

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐIỆN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG

ĐỒ ÁN MÔN HỌC 2

**THIẾT KẾ VÀ THI CÔNG MÔ HÌNH IOT BẢO  
MẬT ĐA TẦNG ỨNG DỤNG TRONG QUY  
TRÌNH XỬ LÝ NƯỚC THẢI CÔNG NGHIỆP**

**NGÀNH HỆ THỐNG NHÚNG VÀ IOT**

Sinh viên: **VÕ MINH THÁI**

MSSV: 22139063

TP. HỒ CHÍ MINH – 01/2026

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐIỆN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT MÁY TÍNH – VIỄN THÔNG

ĐỒ ÁN MÔN HỌC 2

**THIẾT KẾ VÀ THI CÔNG MÔ HÌNH IOT BẢO  
MẬT ĐA TẦNG ỨNG DỤNG TRONG QUY  
TRÌNH XỬ LÝ NƯỚC THẢI CÔNG NGHIỆP**

**NGÀNH HỆ THỐNG NHÚNG VÀ IOT**

Sinh viên: **VÕ MINH THÁI**

MSSV: 22139063

Hướng dẫn: **ThS. TRƯƠNG QUANG PHÚC**

TP. HỒ CHÍ MINH – 01/2026

## LỜI CẢM ƠN

Tôi xin bày tỏ lòng biết ơn sâu sắc đến Thầy Trương Quang Phúc, người đã trực tiếp hướng dẫn và đồng hành cùng tôi trong suốt quá trình thực hiện đề tài Đồ án môn học 2. Với sự tận tâm và tinh thần trách nhiệm cao, Thầy đã dành nhiều thời gian góp ý, chỉ ra những thiếu sót và định hướng lại cách tiếp cận vấn đề, giúp tôi từng bước hoàn thiện đề tài theo đúng mục tiêu và yêu cầu học thuật.

Trong quá trình thực hiện đồ án, do hạn chế về kinh nghiệm thực tế cũng như chưa có cách nhìn toàn diện đối với một hệ thống kỹ thuật hoàn chỉnh, tôi đã gặp không ít sai sót và có những thời điểm làm việc chưa thực sự cẩn trọng. Nhờ sự hướng dẫn kịp thời, những nhận xét thẳng thắn và mang tính xây dựng của Thầy, tôi đã nhận ra các điểm chưa phù hợp trong cách triển khai, từ đó điều chỉnh phương pháp làm việc, bổ sung kiến thức và cải thiện chất lượng nội dung báo cáo.

Thông qua quá trình được Thầy hướng dẫn và chỉnh sửa, tôi không chỉ hoàn thiện đồ án về mặt kỹ thuật mà còn rút ra nhiều bài học quý báu về tư duy hệ thống, cách trình bày báo cáo khoa học và tinh thần nghiêm túc trong nghiên cứu. Đây là những kinh nghiệm có giá trị đối với tôi trong quá trình học tập và làm việc sau này.

Tôi xin chân thành cảm ơn Thầy vì sự hướng dẫn tận tình và những đóng góp quý báu trong suốt quá trình thực hiện đề tài.

# TÓM TẮT

Trong bối cảnh công nghiệp hóa và chuyển đổi số diễn ra mạnh mẽ, việc ứng dụng Internet of Things (IoT) vào các hệ thống xử lý nước thải công nghiệp đang trở thành xu hướng tất yếu nhằm nâng cao hiệu quả giám sát, điều khiển và quản lý vận hành. Tuy nhiên, khi các hệ thống điều khiển được kết nối mạng và cho phép truy cập từ xa, vấn đề bảo mật và kiểm soát truy cập ngày càng trở nên quan trọng, đặc biệt đối với các quy trình có ảnh hưởng trực tiếp đến môi trường và an toàn sản xuất.

Đề tài này tập trung thiết kế và thi công một mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp, với kiến trúc phân lớp rõ ràng gồm tầng thiết bị, tầng gateway và tầng ứng dụng. Trong hệ thống đề xuất, vi điều khiển STM32 đảm nhiệm chức năng thu thập dữ liệu từ các cảm biến pH, độ đục, mực nước và nhiệt độ, đồng thời điều khiển các cơ cấu chấp hành. ESP32 đóng vai trò trung gian truyền thông giữa tầng thiết bị và gateway. Raspberry Pi được sử dụng làm gateway trung tâm, thực hiện xử lý dữ liệu, giao tiếp thông qua giao thức MQTT và kết nối với giao diện Web Dashboard phục vụ giám sát và điều khiển theo thời gian thực.

Điểm nổi bật của mô hình nằm ở cơ chế bảo mật đa tầng, trong đó việc phân quyền và kiểm soát lệnh điều khiển không chỉ được triển khai ở tầng ứng dụng mà còn được tích hợp trực tiếp tại tầng kernel của hệ điều hành Linux trên Raspberry Pi thông qua kernel module và Netfilter. Mỗi lệnh điều khiển đều được kiểm tra, đối chiếu quyền truy cập với cơ sở dữ liệu SQLite trước khi cho phép thực thi, qua đó nâng cao mức độ an toàn và hạn chế truy cập trái phép.

Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, đáp ứng tốt yêu cầu giám sát thời gian thực, độ trễ thấp và cơ chế bảo mật hiệu quả. Mô hình đề xuất có tính khả thi cao, phù hợp cho mục đích nghiên cứu, đào tạo và định hướng mở rộng áp dụng trong các hệ thống xử lý nước thải công nghiệp thực tế.

# MỤC LỤC

DANH MỤC HÌNH . . . . .	vii
DANH MỤC BẢNG . . . . .	x
DANH MỤC CÁC TỪ VIẾT TẮT . . . . .	xi
CHƯƠNG 1. TỔNG QUAN . . . . .	1
1.1 GIỚI THIỆU . . . . .	1
1.2 TÌNH HÌNH NGHIÊN CỨU . . . . .	2
1.2.1 Nghiên cứu ngoài nước . . . . .	2
1.2.2 Nghiên cứu trong nước . . . . .	2
1.3 MỤC TIÊU NGHIÊN CỨU . . . . .	3
1.4 PHƯƠNG PHÁP NGHIÊN CỨU . . . . .	4
1.5 BỐ CỤC ĐỀ TÀI . . . . .	5
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT . . . . .	6
2.1 TỔNG QUAN CÁC QUY TRÌNH XỬ LÝ NƯỚC THẢI CÔNG NGHIỆP ĐIỀN HÌNH . . . . .	6
2.1.1 Giới thiệu một số quy trình . . . . .	6
2.1.2 Các thông số cần giám sát trong thực tế . . . . .	8
2.2 TỔNG QUAN VỀ HỆ ĐIỀU HÀNH RASPBERRY PI . . . . .	9
2.2.1 Giới thiệu về Raspberry Pi OS . . . . .	9
2.2.2 Lịch sử phát triển của Raspberry Pi OS . . . . .	10
2.2.3 Những ưu điểm nổi bật của Raspberry Pi OS . . . . .	10
2.2.4 Nhược điểm của Raspberry Pi OS . . . . .	10
2.3 TỔNG QUAN VỀ FREERTOS . . . . .	10
2.3.1 Giới thiệu về FreeRTOS . . . . .	10
2.3.2 Lịch sử phát triển của FreeRTOS . . . . .	11
2.3.3 Những ưu điểm nổi bật của FreeRTOS . . . . .	11

2.3.4	Nhược điểm của FreeRTOS . . . . .	11
2.4	TỔNG QUAN VỀ LINUX KERNEL VÀ NETFILTER . . . . .	11
2.4.1	Giới thiệu về Linux Kernel . . . . .	11
2.4.2	Kiến trúc tổng quan của Linux Kernel . . . . .	12
2.4.3	Cơ chế Netfilter trong Linux . . . . .	13
2.4.4	Vai trò của Netfilter trong bảo mật hệ thống mạng . . . . .	13
2.5	TỔNG QUAN VỀ FIREBASE . . . . .	13
2.5.1	Giới thiệu chung về Firebase . . . . .	13
2.5.2	Giới thiệu về Firebase Realtime Database . . . . .	14
2.5.3	Giới thiệu về Firebase Authentication . . . . .	14
2.5.4	Giới thiệu về Cloud Firestore . . . . .	14
2.6	TỔNG QUAN VỀ SQLITE . . . . .	14
2.6.1	Giới thiệu về SQLite . . . . .	14
2.6.2	Ưu điểm của SQLite . . . . .	15
2.6.3	Nhược của SQLite . . . . .	15
2.7	TỔNG QUAN VỀ GIAO THỨC MQTT VÀ MOSQUITTO . . . . .	16
2.7.1	Giới thiệu chung về giao thức MQTT . . . . .	16
2.7.2	Mô hình truyền thông Publish/Subscribe của MQTT . . . . .	16
2.7.3	Giới thiệu về MQTT Broker Mosquitto . . . . .	16
2.8	TỔNG QUAN VỀ GIAO THỨC UART . . . . .	17
2.8.1	Nguyên lý hoạt động của UART . . . . .	17
2.8.2	Ưu điểm của UART . . . . .	17
2.8.3	Nhược điểm của UART . . . . .	18
2.9	TỔNG QUAN VỀ RASPBERRY PI 4 . . . . .	18
2.9.1	Cấu tạo của máy tính Raspberry Pi 4 . . . . .	18
2.9.2	Quản lí năng lượng trên Raspberry Pi 4 . . . . .	20
2.10	TỔNG QUAN VỀ CÁC THIẾT BỊ DÙNG TRONG HỆ THỐNG . . . . .	20
2.10.1	Giới thiệu về ESP32 WROOM32 Module . . . . .	20
2.10.2	Giới thiệu về STM32F103C8T6 . . . . .	23

2.10.3	Giới thiệu về cảm biến nhiệt độ DS18B20 . . . . .	27
2.10.4	Giới thiệu về cảm biến PH . . . . .	28
2.10.5	Giới thiệu về cảm biến độ đục . . . . .	29
2.10.6	Giới thiệu về cảm biến khoảng cách HC-SR04 . . . . .	31
<b>CHƯƠNG 3. THIẾT KẾ HỆ THỐNG</b>	. . . . .	<b>34</b>
3.1	YÊU CẦU HỆ THỐNG . . . . .	34
3.2	KIẾN TRÚC ĐỀ XUẤT . . . . .	34
3.3	PHƯƠNG ÁN GIÁM SÁT CHO MÔ HÌNH . . . . .	36
3.3.1	Lựa chọn bể giám sát trong mô hình . . . . .	36
3.3.2	Lựa chọn các thông số cần giám sát . . . . .	36
3.3.3	Lựa chọn thiết bị điều khiển trong mô hình . . . . .	38
3.3.4	Định hướng tích hợp giám sát và điều khiển trong mô hình	38
3.4	ĐẶC TẢ KỸ THUẬT . . . . .	40
3.4.1	Đầu vào của hệ thống . . . . .	40
3.4.2	Đầu ra của hệ thống . . . . .	41
3.4.3	Điều kiện vận hành . . . . .	43
3.5	SƠ ĐỒ KHÓI HỆ THỐNG . . . . .	44
3.6	THIẾT KẾ PHẦN CỨNG . . . . .	46
3.6.1	Thiết kế khói vi điều khiển . . . . .	46
3.6.2	Thiết kế khói cảm biến . . . . .	50
3.6.3	Thiết kế khói điều khiển cơ cấu chấp hành . . . . .	53
3.6.4	Thiết kế khói gateway . . . . .	58
3.6.5	Thiết kế khói nguồn thiết bị . . . . .	59
3.6.6	Thiết kế khói nguồn gateway . . . . .	62
3.7	THIẾT KẾ PHẦN MỀM . . . . .	64
3.7.1	Mô tả luồng hoạt động của hệ thống . . . . .	64
3.7.2	Thiết kế cơ chế bảo bật đa tầng . . . . .	65
3.7.3	Thiết kế phần mềm tầng ứng dụng . . . . .	73
3.7.4	Thiết kế phần mềm tầng gateway . . . . .	91

3.7.5    Thiết kế phần mềm tầng thiết bị . . . . .	96
<b>CHƯƠNG 4. KẾT QUẢ VÀ ĐÁNH GIÁ . . . . .</b>	<b>104</b>
4.1    KẾT QUẢ PHẦN CỨNG . . . . .	105
4.2    KẾT QUẢ PHẦN MỀM . . . . .	107
4.2.1    Kết quả phần mềm tầng ứng dụng . . . . .	107
4.2.2    Kết quả phần mềm tầng gateway . . . . .	112
4.2.3    Kết quả phần mềm tầng thiết bị . . . . .	116
<b>CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN . . . . .</b>	<b>118</b>
5.1    KẾT LUẬN . . . . .	118
5.2    ĐÁNH GIÁ CHUNG . . . . .	118
5.3    HẠN CHẾ . . . . .	119
5.4    HƯỚNG PHÁT TRIỂN . . . . .	119

# DANH MỤC HÌNH

2.1	Sơ đồ quy trình xử lý nước thải công nghệ AAO . . . . .	6
2.2	Sơ đồ quy trình xử lý nước thải theo công nghệ SBR . . . . .	7
2.3	Kiến trúc của Linux Kernel [12] . . . . .	12
2.4	Mô hình Publish/Subscribe của MQTT [20] . . . . .	16
2.5	Giao tiếp giữa hai thiết bị qua UART . . . . .	17
2.6	Sơ đồ chân của Raspberry Pi 4 . . . . .	19
2.7	Sơ đồ nối chân của ESP32 WROOM32 Module 38 chân . . . . .	21
2.8	Sơ đồ chân của STM32F103C8T6 đóng gói kiểu LQFP48 . . . . .	24
2.9	Cảm biến DS18B20 . . . . .	27
2.10	Cảm biến PH . . . . .	28
2.11	Sơ đồ chân board xử lý tín hiệu cảm biến PH . . . . .	29
2.12	Cảm biến độ đục SEN0189 . . . . .	30
2.13	Sơ đồ chân của cảm biến độ đục SEN0189 . . . . .	31
2.14	Cảm biến khoảng cách HC-SR04 . . . . .	32
3.1	Kiến trúc phân tầng của hệ thống . . . . .	35
3.2	Sơ đồ khói của hệ thống . . . . .	44
3.3	Sơ đồ kết nối của khói vi điều khiển . . . . .	47
3.4	Sơ đồ nguyên lý của khói vi điều khiển . . . . .	49
3.5	Vị trí của khói vi điều khiển trên mạch . . . . .	50
3.6	Sơ đồ kết nối của khói cảm biến . . . . .	51
3.7	Sơ đồ nguyên lý của khói cảm biến . . . . .	52
3.8	Vị trí của khói cảm biến trên mạch . . . . .	53
3.9	Sơ đồ kết nối của khói điều khiển cơ cấu chấp hành . . . . .	55
3.10	Sơ đồ nguyên lý khói điều khiển cơ cấu chấp hành . . . . .	57
3.11	Vị trí của khói điều khiển cơ cấu chấp hành trên mạch . . . . .	58
3.12	Sơ đồ kết nối của khói gateway với nguồn . . . . .	59
3.13	Sơ đồ nguyên lý khói nguồn thiết bị . . . . .	60

3.14	Vị trí đặt của khối nguồn thiết bị trên mạch in . . . . .	61
3.15	Sơ nguyên lý toàn bộ hệ thống . . . . .	63
3.16	Sơ đồ mô tả chi tiết hệ thống . . . . .	64
3.17	Cơ chế phân quyền người dùng . . . . .	66
3.18	Lưu đồ thuật toán quá trình bảo mật ở tầng ứng dụng . . . . .	67
3.19	Lưu đồ thuật toán quá trình bảo mật ở tầng ứng dụng - tiếp theo	68
3.20	Lưu đồ thuật toán quá trình bảo mật ở gateway tại kernel-space .	70
3.21	Lưu đồ thuật toán quá trình bảo mật ở gateway tại user-space .	71
3.22	Lưu đồ thuật toán quá trình bảo mật ở tầng thiết bị . . . . .	72
3.23	Lưu đồ thuật toán quá trình xác thực dữ liệu người dùng . . . . .	73
3.24	Lưu đồ thuật toán quá trình gửi dữ liệu xuống gateway . . . . .	74
3.25	Lưu đồ thuật toán quá nhận dữ liệu từ gateway . . . . .	75
3.26	Định hình tổng quan giao diện Admin . . . . .	77
3.27	Định hình tổng quan giao diện điều khiển với quyền Admin .	78
3.28	Lưu đồ minh họa chương trình xử lý lệnh điều khiển với Admin	79
3.29	Định hình tổng quan giao diện User1 . . . . .	80
3.30	Lưu đồ mô tả quá trình điều khiển của người dùng quyền User1	81
3.31	Định hình tổng quan giao diện User2 . . . . .	82
3.32	Lưu đồ mô tả quá trình điều khiển của người dùng quyền User2	83
3.33	Quá trình cấu hình Firebase Authentication hình 1 . . . . .	85
3.34	Cấu trúc của một Firebase Authentication . . . . .	85
3.35	Quá trình cấu hình Firebase Authentication hình 2 . . . . .	86
3.36	Quá trình cấu hình Firebase Authentication hình 3 . . . . .	87
3.37	Cấu trúc của một Firebase Realtime Database . . . . .	88
3.38	Cấu hình Firetore hình 1 . . . . .	89
3.39	Cấu trúc của collection terminal_commands_2 . . . . .	90
3.40	Cấu trúc của collection users . . . . .	91
3.41	Kiến trúc phần mềm trên gateway . . . . .	92
3.42	Lưu đồ cơ chế xử lý dữ liệu cảm biến . . . . .	94

3.43	Lưu đồ cơ chế xử lý dữ liệu điều khiển . . . . .	95
3.44	Lưu đồ thuật toán trên ESP32 . . . . .	97
3.45	Lưu đồ thuật toán trên ESP32 tiếp theo . . . . .	98
3.46	Lưu đồ cấu hình hàm main . . . . .	100
3.47	Lưu đồ thuật toán dùng cho task cảm biến 1 . . . . .	101
3.48	Lưu đồ thuật toán dùng cho task cảm biến 2 . . . . .	102
3.49	Lưu đồ thuật toán phối lệnh điều khiển trên UART2 . . . . .	103
4.1	Mô hình đã hoàn thiện . . . . .	104
4.2	Kết quả thiết kế tầng thiết bị . . . . .	105
4.3	Giao diện đăng nhập . . . . .	107
4.4	Giao diện đăng ký . . . . .	108
4.5	Giao diện giám sát với quyền Admin . . . . .	108
4.6	Giao diện điều khiển với quyền Admin . . . . .	109
4.7	Giao diện giám sát và điều khiển quyền User1 . . . . .	110
4.8	Giao diện hiển thị các biểu đồ dữ liệu quyền User1 . . . . .	111
4.9	Giao diện giám sát và điều khiển quyền User2 . . . . .	111
4.10	Giao diện hiển thị các biểu đồ dữ liệu quyền User2 . . . . .	112
4.11	Load kernel và kiểm tra ban đầu . . . . .	112
4.12	Kiểm thử đầu ra kernel module . . . . .	113
4.13	Kiểm thử vào kernel module . . . . .	114
4.14	Kiểm thử luồng điều khiển . . . . .	114
4.15	Kiểm thử luồng điều khiển tiếp theo . . . . .	114
4.16	Kiểm thử luồng dữ liệu gửi đi . . . . .	115
4.17	Kiểm thử luồng dữ liệu gửi đi tiếp theo . . . . .	115
4.18	Kết quả phần mềm trên tầng thiết bị . . . . .	116
4.19	Kết quả phần mềm trên tầng thiết bị tiếp theo . . . . .	116

# DANH MỤC BẢNG

2.1	Chức năng các nhóm chân trên Raspberry Pi 4 . . . . .	19
2.2	Các tính năng chính của Raspberry Pi 4 . . . . .	20
2.3	Chức năng các nhóm chân trên module ESP32-WROOM-32 . .	22
2.4	Các tính năng nổi bật của module ESP32-WROOM-32 . . . .	22
2.5	Các chế độ quản lý năng lượng tiêu biểu của ESP32 . . . .	23
2.6	Chức năng các nhóm chân trên vi điều khiển STM32F103C8T6	25
2.7	Các tính năng nổi bật của vi điều khiển STM32F103C8T6 . .	26
2.8	Các chế độ quản lý năng lượng tiêu biểu của STM32F103C8T6	26
2.9	Thông số kỹ thuật của cảm biến DS18B20 . . . . .	27
2.10	Chức năng các chân của cảm biến DS18B20 . . . . .	28
2.11	Thông số kỹ thuật cảm biến Gravity: Analog pH Meter V1.1 .	29
2.12	Chân và chức năng của các chân trên cảm biến PH . . . . .	29
2.13	Thông số kỹ thuật cảm biến độ đục Gravity: Analog Turbidity Sensor . . . . .	31
2.14	Chân và chức năng của các chân trên cảm biến độ đục SEN0189	31
2.15	Thông số kỹ thuật cảm biến khoảng cách siêu âm HC-SR04 . .	32
2.16	Chân và chức năng của các chân trên cảm biến HC-SR04 . . . .	32
3.1	Sơ đồ kết nối giữa khối vi điều khiển và nguồn . . . . .	48
3.2	Sơ đồ kết nối nguồn cảm biến . . . . .	51
3.3	Sơ đồ kết nối giữa khối cảm biến và vi điều khiển . . . . .	52
3.4	Sơ đồ kết nối giữa khối cảm biến và vi điều khiển tiếp theo .	52
3.5	Sơ đồ kết nối giữa khối điều khiển cơ cấu chấp hành và vi điều khiển . . . . .	56
3.6	Sơ đồ kết nối của khối gateway . . . . .	59
3.7	Công suất tiêu thụ cực đại của tầng thiết bị . . . . .	61

# DANH MỤC CÁC TỪ VIẾT TẮT

Viết tắt	Ý nghĩa
AAO	Anaerobic–Anoxic–Oxic (Quy trình xử lý nước thải AAO)
ADC	Analog-to-Digital Converter (Bộ chuyển đổi tương tự sang số)
API	Application Programming Interface (Giao diện lập trình ứng dụng)
BOD	Biochemical Oxygen Demand (Nhu cầu oxy sinh hóa)
COD	Chemical Oxygen Demand (Nhu cầu oxy hóa học)
CPU	Central Processing Unit (Bộ xử lý trung tâm)
DMA	Direct Memory Access (Truy cập bộ nhớ trực tiếp)
ESP32	ESP32 Microcontroller (Vi điều khiển ESP32)
FreeRTOS	Free Real-Time Operating System (Hệ điều hành thời gian thực)
GPIO	General Purpose Input/Output (Chân vào/ra đa dụng)
I2C	Inter-Integrated Circuit (Chuẩn truyền thông I2C)
IEC	International Electrotechnical Commission (Ủy ban Kỹ thuật Điện Quốc tế)
IIoT	Industrial Internet of Things (Internet vạn vật công nghiệp)
IoT	Internet of Things (Internet vạn vật)
ISA	International Society of Automation (Tổ chức Tự động hóa Quốc tế)
MES	Manufacturing Execution System (Hệ thống điều hành sản xuất)
MQTT	Message Queuing Telemetry Transport (Giao thức truyền thông MQTT)
Netfilter	Netfilter Framework (Khung lọc gói trong nhân Linux)
OPC-UA	Open Platform Communications – Unified Architecture (Chuẩn truyền thông OPC-UA)
PCB	Printed Circuit Board (Bảng mạch in)
pH	Potential of Hydrogen (Độ axit/kiềm)

<b>Viết tắt</b>	<b>Ý nghĩa</b>
PLC	Programmable Logic Controller (Bộ điều khiển logic lập trình)
PWM	Pulse Width Modulation (Điều chế độ rộng xung)
RTOS	Real-Time Operating System (Hệ điều hành thời gian thực)
SCADA	Supervisory Control and Data Acquisition (Hệ thống điều khiển giám sát)
SBR	Sequencing Batch Reactor (Bể phản ứng theo mẻ tuần tự)
SPI	Serial Peripheral Interface (Chuẩn truyền thông SPI)
SQLite	Lightweight SQL Embedded Database (Cơ sở dữ liệu nhúng SQLite)
TCP/IP	Transmission Control Protocol / Internet Protocol (Bộ giao thức TCP/IP)
TSS	Total Suspended Solids (Chất rắn lơ lửng)
UART	Universal Asynchronous Receiver Transmitter (Chuẩn truyền thông nối tiếp không đồng bộ)

# CHƯƠNG 1. TỔNG QUAN

## 1.1 GIỚI THIỆU

Trong bối cảnh công nghiệp hóa và hiện đại hóa diễn ra mạnh mẽ, vấn đề xử lý nước thải công nghiệp đang ngày càng được quan tâm do ảnh hưởng trực tiếp đến môi trường, sức khỏe con người và sự phát triển bền vững của xã hội. Các hệ thống xử lý nước thải hiện đại không chỉ yêu cầu khả năng vận hành ổn định mà còn đòi hỏi khả năng giám sát liên tục, điều khiển linh hoạt và đảm bảo an toàn khi kết nối mạng [1].

Trong thực tế, nhiều hệ thống xử lý nước thải công nghiệp vẫn sử dụng các giải pháp truyền thống như PLC/SCADA. Mặc dù các hệ thống này có độ tin cậy cao và phù hợp cho môi trường công nghiệp, nhưng khi mở rộng sang mô hình giám sát và điều khiển từ xa thông qua Internet, các vấn đề về phân quyền truy cập, bảo mật dữ liệu và an toàn hệ thống bắt đầu bộc lộ nhiều hạn chế [2]. Việc thiếu cơ chế bảo mật đa tầng có thể dẫn đến nguy cơ bị truy cập trái phép, thao tác sai quy trình hoặc tấn công mạng vào hệ thống điều khiển.

Sự phát triển của Internet of Things (IoT) đã mở ra nhiều cơ hội mới trong lĩnh vực giám sát và điều khiển hệ thống xử lý nước thải. IoT cho phép kết nối các cảm biến, thiết bị chấp hành và hệ thống điều khiển thông qua mạng Internet, từ đó nâng cao khả năng giám sát theo thời gian thực, thu thập dữ liệu và điều khiển từ xa. Tuy nhiên, trong nhiều mô hình IoT hiện nay, vấn đề phân quyền người dùng và bảo mật đa tầng vẫn chưa được chú trọng đúng mức, đặc biệt khi hệ thống có nhiều vai trò người dùng và nhiều tầng kiến trúc khác nhau.

Xuất phát từ những yêu cầu thực tiễn đó, đề tài “Thiết kế và thi công mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp” được thực hiện với mục tiêu xây dựng một mô hình IoT gồm nhiều tầng chức năng, trong đó yếu tố bảo mật và phân quyền được tích hợp xuyên suốt toàn bộ hệ thống.

## **1.2 TÌNH HÌNH NGHIÊN CỨU**

### **1.2.1 Nghiên cứu ngoài nước**

Trên thế giới, các hệ thống xử lý nước thải công nghiệp ngày càng được tích hợp với các giải pháp giám sát và điều khiển dựa trên Internet of Things (IoT) nhằm nâng cao hiệu quả vận hành và giám sự phụ thuộc vào con người. Tuy nhiên, khi các hệ thống điều khiển công nghiệp được kết nối mạng và cho phép giám sát, điều khiển từ xa, vấn đề bảo mật và an toàn hệ thống trở thành mối quan tâm đặc biệt quan trọng, nhất là đối với các hạ tầng liên quan trực tiếp đến môi trường và sức khỏe cộng đồng [3].

Theo [4] đã chỉ ra rằng các hệ thống IoT công nghiệp trong lĩnh vực môi trường, bao gồm xử lý nước thải, có nguy cơ cao bị truy cập trái phép, giả mạo lệnh điều khiển hoặc can thiệp vào dữ liệu vận hành nếu không được thiết kế với cơ chế bảo mật phù hợp. Các sự cố bảo mật trong các hệ thống này không chỉ gây gián đoạn quá trình xử lý mà còn có thể dẫn đến xả thải không kiểm soát, gây tác động nghiêm trọng đến môi trường.

Do đó, các nghiên cứu ngoài nước nhấn mạnh nhu cầu xây dựng các hệ thống IoT công nghiệp với cơ chế bảo mật chặt chẽ, bao gồm xác thực người dùng, phân quyền truy cập, kiểm soát lệnh điều khiển và giám sát an ninh hệ thống. Tuy nhiên, phần lớn các giải pháp được đề xuất có kiến trúc phức tạp và chi phí triển khai cao, đặt ra yêu cầu cần có những mô hình IoT bảo mật, dễ triển khai và phù hợp hơn cho các hệ thống xử lý nước thải trong thực tế [5].

### **1.2.2 Nghiên cứu trong nước**

Trong những năm gần đây, cùng với xu hướng chuyển đổi số quốc gia, lĩnh vực giám sát môi trường tại Việt Nam đã có nhiều bước phát triển đáng kể, đặc biệt là trong việc ứng dụng công nghệ số và Internet of Things (IoT). Theo nghiên cứu [6], việc ứng dụng IoT, dữ liệu lớn và các nền tảng số đã góp phần nâng cao hiệu quả thu thập, quản lý và khai thác dữ liệu môi trường, trong đó có lĩnh vực giám sát và xử lý nước thải công nghiệp.

Nhiều hệ thống giám sát môi trường hiện nay đã được triển khai theo hướng tự động hóa, cho phép thu thập dữ liệu cảm biến liên tục, giám sát và hỗ trợ điều khiển quy trình vận hành. Dữ liệu đo được truyền về trung tâm thông qua các nền tảng mạng, sau đó được lưu trữ và hiển thị nhằm phục vụ công tác quản lý và giám sát theo thời gian thực. Các mô hình này góp phần giảm sự phụ thuộc vào phương pháp quan trắc thủ công, đồng thời nâng cao tính liên tục và độ tin cậy của dữ liệu môi trường [6].

Nhưng thực tế cho thấy phần lớn các hệ thống giám sát môi trường tại Việt Nam hiện nay vẫn chủ yếu tập trung vào chức năng thu thập và hiển thị dữ liệu. Các yêu cầu về bảo mật hệ thống, đặc biệt là bảo mật truy cập, xác thực người dùng và kiểm soát các thao tác điều khiển từ xa, vẫn chưa được triển khai một cách đầy đủ và đồng bộ. Khi hệ thống được kết nối mạng hoặc tích hợp với các nền tảng số, nguy cơ bị truy cập trái phép, sai lệch dữ liệu hoặc can thiệp không mong muốn vào quá trình vận hành có thể phát sinh nếu không có cơ chế bảo mật phù hợp [6].

Có thể thấy, các giải pháp chuyển đổi số trong giám sát môi trường tại Việt Nam vẫn đang trong giai đoạn hoàn thiện, chủ yếu mang tính thử nghiệm hoặc triển khai cục bộ. Việc xây dựng các mô hình IoT có khả năng mở rộng, đảm bảo an ninh – an toàn thông tin và phù hợp với yêu cầu vận hành thực tế trong môi trường công nghiệp vẫn là một thách thức lớn. Điều này đặt ra nhu cầu nghiên cứu và phát triển các mô hình IoT bảo mật đa tầng, có cơ chế phân quyền rõ ràng, nhằm đáp ứng yêu cầu giám sát và điều khiển hệ thống xử lý nước thải trong bối cảnh chuyển đổi số hiện nay.

### **1.3 MỤC TIÊU NGHIÊN CỨU**

Mục tiêu của đề tài là nghiên cứu, thiết kế và triển khai một mô hình Internet of Things (IoT) có kiến trúc phân tầng và cơ chế bảo mật đa tầng, ứng dụng trong bài toán giám sát và điều khiển quy trình xử lý nước thải công nghiệp. Trên cơ sở đó, đề tài hướng đến các mục tiêu cụ thể sau:

- Xây dựng một kiến trúc hệ thống IoT tương đối hoàn chỉnh, đảm bảo tính rõ

ràng trong phân chia chức năng, thuận lợi cho việc quản lý, mở rộng và tích hợp trong môi trường công nghiệp.

- Triển khai mô hình trên bộ kit mô phỏng, phản ánh các đặc trưng cơ bản của quy trình xử lý nước thải, bao gồm thu thập dữ liệu cảm biến, giám sát trạng thái hệ thống và thực hiện điều khiển từ xa.
- Nghiên cứu và áp dụng các cơ chế bảo mật phù hợp tại từng tầng của hệ thống, bao gồm xác thực người dùng, phân quyền truy cập, bảo vệ dữ liệu truyền thông và kiểm soát các thao tác điều khiển, nhằm hạn chế tối đa các nguy cơ mất an toàn thông tin khi hệ thống được kết nối mạng.
- Xây dựng giao diện giám sát và điều khiển trực quan, hỗ trợ nhiều vai trò người dùng khác nhau, qua đó đánh giá khả năng ứng dụng thực tế của mô hình IoT bảo mật đa tầng trong các hệ thống xử lý nước thải công nghiệp.

## 1.4 PHƯƠNG PHÁP NGHIÊN CỨU

Để đạt được các mục tiêu đề ra, đề tài được thực hiện dựa trên sự kết hợp giữa phương pháp nghiên cứu lý thuyết và phương pháp nghiên cứu thực nghiệm, tập trung vào thiết kế, triển khai và đánh giá một mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp.

Trước hết, tiến hành nghiên cứu tài liệu liên quan đến các hệ thống IoT công nghiệp, kiến trúc phân tầng, các giải pháp bảo mật trong hệ thống điều khiển và giám sát từ xa, đặc biệt là các nghiên cứu trong lĩnh vực xử lý nước thải và giám sát môi trường. Việc tổng hợp và phân tích các tài liệu trong và ngoài nước giúp xác định những hạn chế của các mô hình hiện có, đồng thời làm cơ sở để xuất kiến trúc hệ thống phù hợp với mục tiêu của đề tài.

Tiếp theo, đề tài áp dụng phương pháp phân tích – thiết kế hệ thống để xây dựng kiến trúc IoT phân tầng gồm tầng thiết bị, tầng gateway và tầng ứng dụng. Trên cơ sở đó, các chức năng chính của từng tầng được xác định rõ ràng, bao gồm thu thập dữ liệu cảm biến, điều khiển thiết bị chấp hành, truyền thông dữ liệu, lưu trữ, phân quyền truy cập và bảo mật hệ thống.

Sau khi hoàn thiện thiết kế, phương pháp thực nghiệm được sử dụng để triển khai mô hình trên bộ kit mô phỏng. Các phần cứng như vi điều khiển, cảm biến và thiết bị chấp hành được tích hợp với gateway và nền tảng lưu trữ đám mây. Đồng thời, các cơ chế bảo mật như xác thực người dùng, phân quyền truy cập và kiểm soát lệnh điều khiển được triển khai và kiểm tra trong quá trình vận hành.

Cuối cùng, đề tài tiến hành đánh giá và phân tích kết quả dựa trên khả năng giám sát, điều khiển hệ thống theo thời gian thực, mức độ ổn định của truyền thông, cũng như hiệu quả của các cơ chế bảo mật được áp dụng. Kết quả thực nghiệm được so sánh với các mục tiêu ban đầu nhằm đánh giá tính khả thi và khả năng ứng dụng của mô hình trong thực tế.

## 1.5 BỘ CỤC ĐỀ TÀI

Chương 1 trình bày bối cảnh và lý do lựa chọn đề tài, giới thiệu tổng quan đề tài và xu hướng ứng dụng IoT trong giám sát – điều khiển.

Chương 2 trình bày các cơ sở lý thuyết liên quan đến đề tài, bao gồm các công nghệ truyền thông sử dụng, các thiết phần cứng, phần mềm dùng cho hệ thống.

Chương 3 trình bày các yêu cầu, đặc tả kỹ thuật, sơ đồ khái niệm, quá trình thiết kế phần cứng và phần mềm.

Chương 4 nêu kết quả thực hiện trên hệ thống.

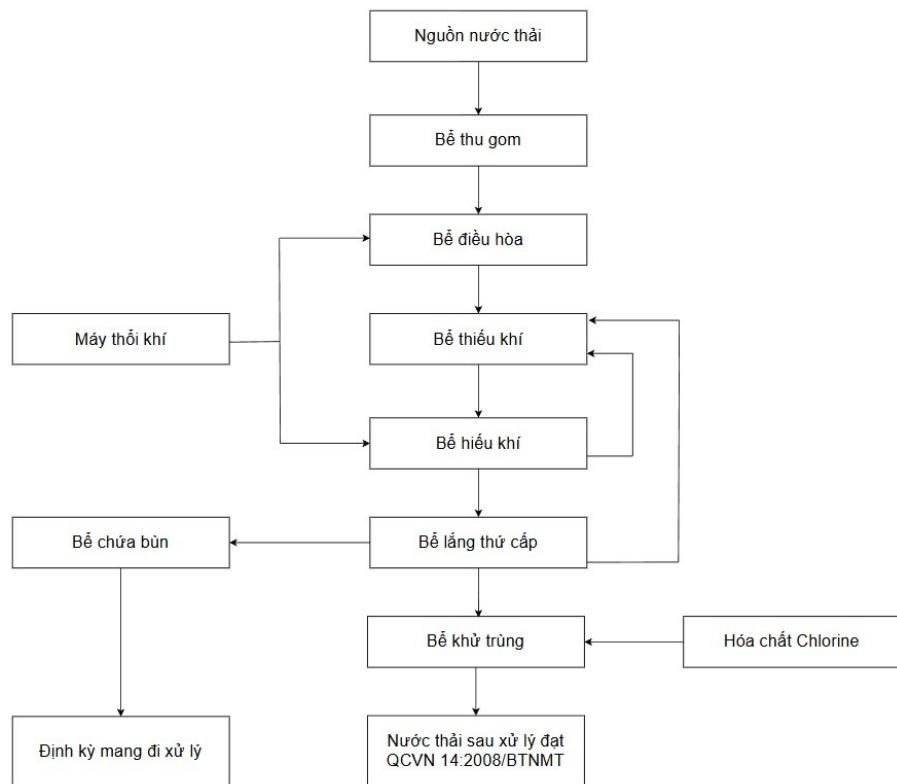
Chương 5 nhận xét các yêu cầu đã đáp ứng được, những hạn chế của hệ thống và đề xuất ra các giải pháp cũng như hướng phát triển cho tương lai.

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

## 2.1 TỔNG QUAN CÁC QUY TRÌNH XỬ LÝ NƯỚC THẢI CÔNG NGHIỆP ĐIỀN HÌNH

### 2.1.1 Giới thiệu một số quy trình

Các hệ thống xử lý nước thải trong công nghiệp hiện đại có nhiều mô hình và quy trình khác nhau, tùy thuộc vào loại hình sản xuất, lưu lượng nước thải và yêu cầu chất lượng nước đầu ra. Trên cơ sở đó, để minh họa cho sự đa dạng của các hệ thống xử lý nước thải trong thực tế ta cùng đi khảo sát một vài quy trình cụ thể:



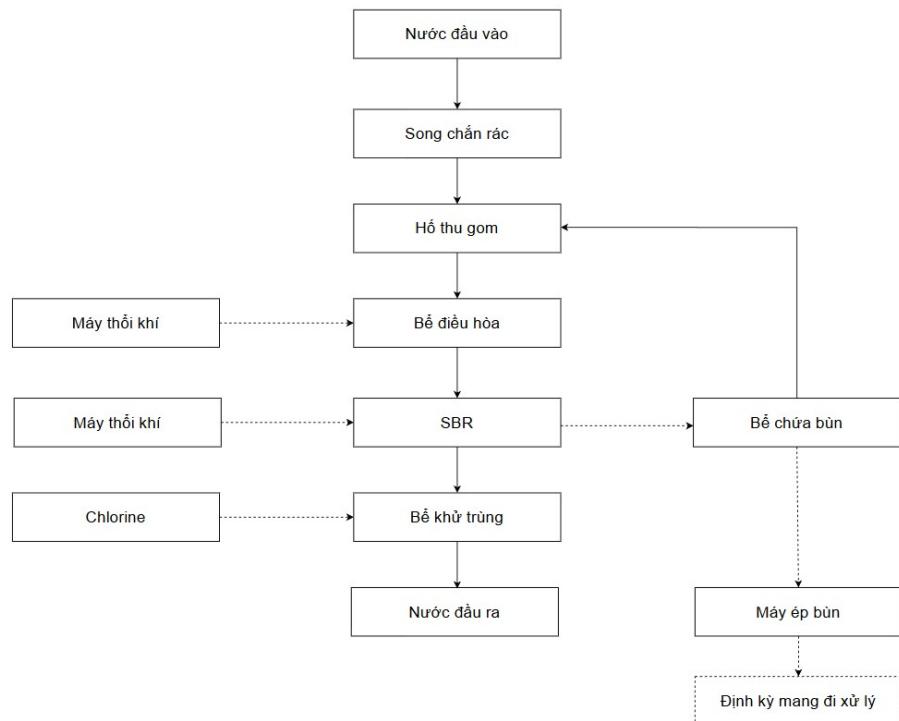
Hình 2.1: Sơ đồ quy trình xử lý nước thải công nghệ AAO

Hình minh họa trên được tham khảo và vẽ lại tại [7], đây là một quy trình xử lý nước thải công nghiệp điển hình theo công nghệ AAO (Anaerobic – Anoxic – Oxic), bao gồm các công đoạn chính như thu gom nước thải, điều hòa lưu lượng, xử lý sinh học, lắng bùn, khử trùng và xả thải đạt quy chuẩn môi trường. Quy trình

này được tổng hợp và tham khảo từ các mô hình xử lý nước thải công nghiệp đang được áp dụng phổ biến trong thực tế [7].

Theo [7] Công nghệ AAO là một phương pháp xử lý sinh học kết hợp ba vùng phản ứng liên tiếp gồm khí khí, thiếu khí và hiếu khí. Nhờ sự phân tách các vùng phản ứng với điều kiện môi trường khác nhau, công nghệ AAO cho phép xử lý hiệu quả các chất hữu cơ, hợp chất nitơ và photpho trong nước thải, đồng thời đảm bảo chất lượng nước đầu ra đáp ứng các quy chuẩn môi trường hiện hành. Công nghệ này hiện được ứng dụng rộng rãi trong các hệ thống xử lý nước thải công nghiệp và đô thị nhờ tính ổn định và khả năng mở rộng cao.

Bên cạnh công nghệ AAO, công nghệ SBR cũng được ứng dụng phổ biến trong xử lý nước thải công nghiệp nhờ tính linh hoạt trong vận hành; Hình dưới sẽ minh họa quy trình xử lý nước thải theo công nghệ SBR.



Hình 2.2: Sơ đồ quy trình xử lý nước thải theo công nghệ SBR

Hình trên được tham khảo tại [7], mục đích là trình bày một quy trình xử lý nước thải công nghiệp điển hình theo công nghệ SBR (Sequencing Batch Reactor). Quy trình này bao gồm các giai đoạn chính như nạp nước thải, phản ứng sinh học,

lắng, xả nước và chờ, được thực hiện tuần tự trong cùng một bể xử lý. Công nghệ SBR được tổng hợp và tham khảo từ các mô hình xử lý nước thải công nghiệp đang được áp dụng phổ biến trong thực tế [7].

Theo [7], công nghệ SBR là một phương pháp xử lý sinh học theo mẻ, trong đó các quá trình xử lý diễn ra theo chu trình thời gian thay vì dòng chảy liên tục. Nhờ khả năng điều chỉnh linh hoạt thời gian của từng giai đoạn xử lý, công nghệ SBR cho phép nâng cao hiệu quả xử lý các chất hữu cơ và chất dinh dưỡng trong nước thải. Bên cạnh đó, cấu trúc hệ thống gọn nhẹ và dễ điều khiển giúp công nghệ SBR phù hợp với các hệ thống xử lý nước thải có quy mô vừa và nhỏ, cũng như các mô hình nghiên cứu và mô phỏng.

Qua việc khảo sát hai quy trình xử lý nước thải công nghiệp tiêu biểu là công nghệ AAO và công nghệ SBR, có thể thấy rằng mặc dù khác nhau về cấu trúc và phương thức vận hành, cả hai đều hướng tới mục tiêu chung là xử lý hiệu quả các chất ô nhiễm và đảm bảo chất lượng nước đầu ra đạt quy chuẩn môi trường. Các quy trình này đều bao gồm những công đoạn then chốt cần được giám sát chặt chẽ nhằm đảm bảo hệ thống vận hành ổn định và an toàn.

Việc tổng quan các quy trình xử lý nước thải điển hình giúp làm rõ bối cảnh ứng dụng của các hệ thống giám sát và điều khiển trong công nghiệp, đồng thời tạo cơ sở để lựa chọn các công đoạn và thông số phù hợp cho việc xây dựng mô hình giám sát – điều khiển bằng IoT trong các phần tiếp theo của báo cáo.

### 2.1.2 Các thông số cần giám sát trong thực tế

Trong thực tế vận hành các hệ thống xử lý nước thải công nghiệp, việc giám sát các thông số đặc trưng của nước thải là yêu cầu quan trọng nhằm đánh giá hiệu quả xử lý, đảm bảo an toàn vận hành và đáp ứng các quy chuẩn môi trường. Theo [8], các thông số giám sát thường bao gồm nhóm thông số vật lý, hóa học và các chỉ tiêu phản ánh mức độ ô nhiễm.

Các thông số vật lý cơ bản cần được theo dõi gồm lưu lượng, mực nước và nhiệt độ. Những thông số này giúp kiểm soát tải trọng hệ thống, phát hiện tình trạng quá

tải hoặc tràn bể, đồng thời ảnh hưởng trực tiếp đến hiệu suất của các quá trình xử lý sinh học.

Nhóm thông số hóa học đóng vai trò then chốt trong việc đánh giá chất lượng nước thải, trong đó pH là thông số bắt buộc phải giám sát do ảnh hưởng trực tiếp đến hoạt động của vi sinh vật và độ bền thiết bị. Bên cạnh đó, các chỉ tiêu như COD, BOD phản ánh mức độ ô nhiễm hữu cơ và thường được sử dụng để đánh giá hiệu quả xử lý tổng thể của hệ thống.

Ngoài ra, các thông số như TSS và độ đục được sử dụng để đánh giá hàm lượng chất rắn lơ lửng trong nước thải, có ý nghĩa quan trọng trong kiểm soát quá trình lắng và chất lượng nước đầu ra. Đối với một số hệ thống, các chỉ tiêu về nitơ và photpho cũng được giám sát nhằm kiểm soát hiện tượng phú dưỡng và đảm bảo yêu cầu xả thải [8].

Từ thực tế triển khai cho thấy, việc lựa chọn các thông số giám sát phụ thuộc vào đặc điểm công nghệ xử lý, yêu cầu pháp lý và điều kiện vận hành cụ thể của từng hệ thống xử lý nước thải công nghiệp.

Tóm lại, các nội dung trên cho thấy vai trò quan trọng của việc giám sát thông số trong hệ thống xử lý nước thải công nghiệp, đồng thời là cơ sở để lựa chọn và triển khai các giải pháp giám sát phù hợp trong các mô hình nghiên cứu và ứng dụng cụ thể.

## 2.2 TỔNG QUAN VỀ HỆ ĐIỀU HÀNH RASPBERRY PI

### 2.2.1 Giới thiệu về Raspberry Pi OS

Raspberry Pi OS là hệ điều hành dựa trên Linux được phát triển và duy trì bởi Raspberry Pi Foundation, thiết kế tối ưu cho các dòng máy tính nhúng Raspberry Pi. Hệ điều hành này cung cấp môi trường ổn định để phát triển và triển khai các ứng dụng nhúng, mạng và Internet of Things (IoT)[9].

### **2.2.2 Lịch sử phát triển của Raspberry Pi OS**

Raspberry Pi OS được giới thiệu lần đầu vào năm 2012 cùng với sự ra đời của Raspberry Pi, với mục tiêu cung cấp một hệ điều hành nhẹ, dễ sử dụng và phù hợp cho giáo dục cũng như các ứng dụng nhúng. Trải qua quá trình phát triển, hệ điều hành này liên tục được cập nhật để cải thiện hiệu năng, tăng cường bảo mật và mở rộng khả năng hỗ trợ phần cứng mới, trở thành nền tảng phổ biến nhất cho các ứng dụng trên Raspberry Pi [9].

### **2.2.3 Những ưu điểm nổi bật của Raspberry Pi OS**

Raspberry Pi OS có ưu điểm nổi bật là tính ổn định cao, khả năng tương thích tốt với phần cứng Raspberry Pi và hệ sinh thái phần mềm phong phú. Hệ điều hành hỗ trợ đầy đủ các dịch vụ mạng, cho phép triển khai các ứng dụng đa nhiệm, xử lý dữ liệu và giao tiếp với các thiết bị ngoại vi. Ngoài ra, do dựa trên nền tảng Linux, Raspberry Pi OS cho phép người dùng tùy biến sâu và triển khai các cơ chế bảo mật phù hợp với yêu cầu hệ thống.

### **2.2.4 Nhược điểm của Raspberry Pi OS**

Mặc dù phù hợp với nhiều ứng dụng nhúng, Raspberry Pi OS vẫn tồn tại một số hạn chế như chưa đảm bảo tính thời gian thực cứng và yêu cầu tài nguyên cao hơn so với các hệ điều hành nhúng chuyên dụng. Do đó, trong các hệ thống điều khiển đòi hỏi độ đáp ứng thời gian thực cao, Raspberry Pi OS thường được sử dụng kết hợp với các vi điều khiển hoặc giải pháp phân tầng xử lý.

## **2.3 TỔNG QUAN VỀ FREERTOS**

### **2.3.1 Giới thiệu về FreeRTOS**

FreeRTOS là một hệ điều hành thời gian thực (RTOS) mã nguồn mở được thiết kế dành cho các hệ thống nhúng và vi điều khiển có tài nguyên hạn chế. FreeRTOS cung cấp cơ chế lập lịch đa nhiệm theo ưu tiên, cho phép các tác vụ (task) được thực thi một cách có kiểm soát và đáp ứng yêu cầu thời gian thực. Nhờ thiết kế gọn

nhẹ và khả năng cấu hình linh hoạt, FreeRTOS được sử dụng rộng rãi trong các hệ thống IoT, điều khiển và giám sát công nghiệp [10].

### **2.3.2 Lịch sử phát triển của FreeRTOS**

FreeRTOS được phát triển lần đầu vào năm 2003 bởi Richard Barry với mục tiêu cung cấp một RTOS đơn giản, dễ tích hợp cho các vi điều khiển phổ biến. Trải qua quá trình phát triển và hoàn thiện, FreeRTOS hiện nay được Amazon duy trì và hỗ trợ, với khả năng chạy trên hàng trăm kiến trúc vi điều khiển khác nhau. FreeRTOS đã trở thành một trong những RTOS được sử dụng phổ biến nhất trong lĩnh vực hệ thống nhúng [11].

### **2.3.3 Những ưu điểm nổi bật của FreeRTOS**

FreeRTOS có ưu điểm nổi bật là nhẹ, dễ triển khai và hiệu quả. Hệ điều hành cung cấp các cơ chế quan trọng như quản lý task, semaphore, mutex, queue và timer, giúp lập trình viên xây dựng các ứng dụng đa nhiệm ổn định và dễ mở rộng. Bên cạnh đó, FreeRTOS có tính di động cao, tài liệu phong phú và cộng đồng hỗ trợ rộng lớn, phù hợp cho các ứng dụng nhúng và IoT yêu cầu đáp ứng thời gian thực.

### **2.3.4 Nhược điểm của FreeRTOS**

Tuy có nhiều ưu điểm, FreeRTOS vẫn tồn tại một số hạn chế như không cung cấp đầy đủ các dịch vụ của một hệ điều hành tổng quát (ví dụ: hệ thống file, quản lý tiến trình phức tạp) và yêu cầu lập trình viên có kiến thức tốt về lập trình nhúng để tránh lỗi đồng bộ hoặc tràn bộ nhớ.

## **2.4 TỔNG QUAN VỀ LINUX KERNEL VÀ NETFILTER**

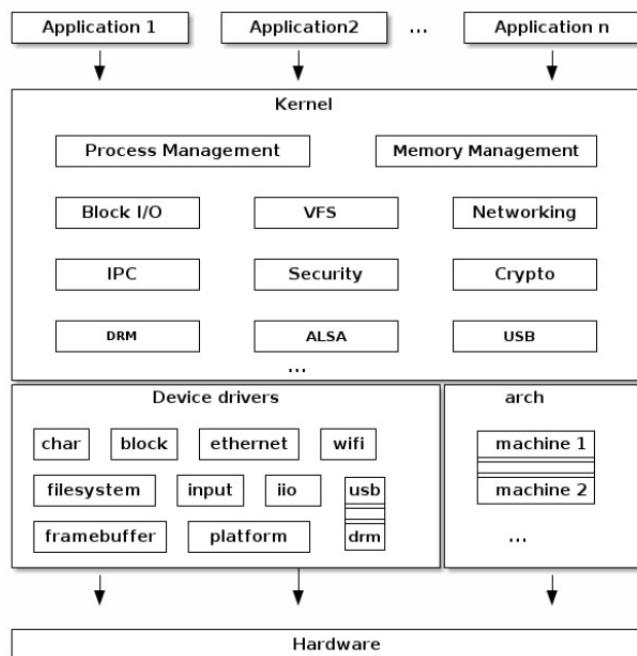
### **2.4.1 Giới thiệu về Linux Kernel**

Linux Kernel là nhân của hệ điều hành Linux, chịu trách nhiệm quản lý và điều phối các tài nguyên phần cứng của hệ thống như CPU, bộ nhớ, thiết bị ngoại vi và

giao tiếp mạng. Kernel đóng vai trò trung gian giữa phần cứng và các ứng dụng ở không gian người dùng (user space), đảm bảo hệ thống hoạt động ổn định, an toàn và hiệu quả. Trong các hệ thống nhúng và gateway IoT, Linux Kernel cho phép triển khai các cơ chế xử lý ở tầng thấp, đặc biệt là các chức năng liên quan đến mạng và bảo mật [12].

#### 2.4.2 Kiến trúc tổng quan của Linux Kernel

Về mặt kiến trúc, Linux được tổ chức theo mô hình kernel space và user space. Các ứng dụng người dùng hoạt động trong user space và không được phép truy cập trực tiếp vào phần cứng. Mọi thao tác liên quan đến tài nguyên hệ thống đều phải thông qua kernel space. Linux Kernel bao gồm nhiều phân hệ quan trọng như quản lý tiến trình, quản lý bộ nhớ, hệ thống tập tin, trình điều khiển thiết bị và ngăn xếp mạng (network stack). Ngăn xếp mạng là nền tảng để triển khai các cơ chế xử lý và kiểm soát gói tin, trong đó Netfilter là một thành phần cốt lõi [12].



Hình 2.3: Kiến trúc của Linux Kernel [12]

### **2.4.3 Cơ chế Netfilter trong Linux**

Netfilter là một framework xử lý gói tin trong Linux Kernel, cho phép can thiệp vào quá trình xử lý các gói dữ liệu mạng tại các thời điểm khác nhau của ngăn xếp mạng. Netfilter cung cấp các hook tại các điểm quan trọng như khi gói tin vào hệ thống, chuyển tiếp, xử lý cục bộ hoặc rời khỏi hệ thống. [13]. Thông qua các hook này, kernel có thể thực hiện các chức năng như lọc gói tin, chuyển tiếp, thay đổi hoặc chặn gói dữ liệu.

### **2.4.4 Vai trò của Netfilter trong bảo mật hệ thống mạng**

Trong các hệ thống có kết nối mạng, Netfilter đóng vai trò quan trọng trong việc kiểm soát truy cập và bảo mật hệ thống. Việc xử lý gói tin ngay tại tầng kernel giúp phát hiện và ngăn chặn các truy cập trái phép trước khi chúng ảnh hưởng đến các dịch vụ ở tầng ứng dụng.

Thông qua Netfilter, hệ thống có thể triển khai các chính sách bảo mật như chỉ cho phép lưu thông các gói tin sử dụng cổng mạng hợp lệ, chặn các kết nối không mong muốn và giám sát lưu lượng mạng. Cách tiếp cận này góp phần xây dựng cơ chế bảo mật đa tầng, trong đó lớp bảo vệ ở tầng kernel giúp tăng mức độ an toàn cho toàn bộ hệ thống khi hoạt động trong môi trường mạng mở.

## **2.5 TỔNG QUAN VỀ FIREBASE**

### **2.5.1 Giới thiệu chung về Firebase**

Firebase là một nền tảng phát triển ứng dụng do Google cung cấp, hoạt động theo mô hình Backend-as-a-Service (BaaS), cho phép các nhà phát triển xây dựng và triển khai ứng dụng mà không cần tự quản lý hạ tầng máy chủ. Firebase cung cấp một tập hợp các dịch vụ backend được tích hợp sẵn, hỗ trợ cả ứng dụng web và ứng dụng di động [14].

Các dịch vụ cốt lõi của Firebase bao gồm xác thực người dùng (Authentication), cơ sở dữ liệu thời gian thực (Realtime Database), cơ sở dữ liệu tài liệu (Cloud Firestore), lưu trữ dữ liệu (Cloud Storage) và hệ thống quản lý bảo mật thông qua

các luật truy cập. Nhờ kiến trúc dựa trên đám mây, Firebase cho phép ứng dụng dễ dàng mở rộng, đồng bộ dữ liệu nhanh chóng và duy trì kết nối liên tục giữa client và server [14].

### **2.5.2 Giới thiệu về Firebase Realtime Database**

Firebase Realtime Database là một hệ cơ sở dữ liệu NoSQL dạng cây, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa các thiết bị và ứng dụng. Khi dữ liệu tại một nút thay đổi, các client đang kết nối sẽ nhận được cập nhật ngay lập tức mà không cần truy vấn lại [15].

### **2.5.3 Giới thiệu về Firebase Authentication**

Firebase Authentication là dịch vụ xác thực người dùng, hỗ trợ quản lý danh tính và quyền truy cập vào hệ thống. Dịch vụ này cung cấp cơ chế đăng nhập an toàn thông qua tài khoản và mật khẩu, đồng thời gán cho mỗi người dùng một mã định danh duy nhất (UID) [16].

### **2.5.4 Giới thiệu về Cloud Firestore**

Cloud Firestore là một hệ cơ sở dữ liệu NoSQL dạng tài liệu (document-based), được thiết kế để lưu trữ dữ liệu có cấu trúc rõ ràng và ít thay đổi theo thời gian. Firestore hỗ trợ truy vấn linh hoạt, khả năng mở rộng cao và cơ chế bảo mật dựa trên luật truy cập (rules) [17].

## **2.6 TỔNG QUAN VỀ SQLITE**

### **2.6.1 Giới thiệu về SQLite**

SQLite là một hệ quản trị cơ sở dữ liệu quan hệ nhẹ (lightweight relational database), hoạt động theo mô hình embedded database, trong đó toàn bộ cơ sở dữ liệu được lưu trữ trong một tệp duy nhất trên hệ thống. Không giống các hệ quản trị cơ sở dữ liệu truyền thống, SQLite không yêu cầu máy chủ riêng và được tích hợp trực tiếp vào ứng dụng thông qua thư viện liên kết.

SQLite hỗ trợ đầy đủ các thao tác cơ bản của ngôn ngữ SQL như tạo bảng, truy vấn, cập nhật và xóa dữ liệu. Nhờ kiến trúc đơn giản, dễ triển khai và tiêu tốn ít tài nguyên, SQLite được sử dụng rộng rãi trong các hệ thống nhúng, thiết bị IoT và các ứng dụng cần lưu trữ dữ liệu cục bộ [18].

### 2.6.2 Ưu điểm của SQLite

SQLite sở hữu nhiều ưu điểm nổi bật, đặc biệt phù hợp với các hệ thống nhúng:

- Không cần máy chủ: SQLite hoạt động hoàn toàn độc lập, giúp giảm độ phức tạp khi triển khai hệ thống.
- Nhẹ và hiệu quả: Dung lượng nhỏ, tiêu thụ ít bộ nhớ và tài nguyên CPU.
- Dễ tích hợp: Có thể nhúng trực tiếp vào ứng dụng thông qua thư viện, hỗ trợ nhiều ngôn ngữ lập trình.
- Phù hợp lưu trữ cục bộ: Thích hợp cho việc lưu lịch sử, trạng thái và dữ liệu trung gian trong các hệ thống IoT.

### 2.6.3 Nhược điểm của SQLite

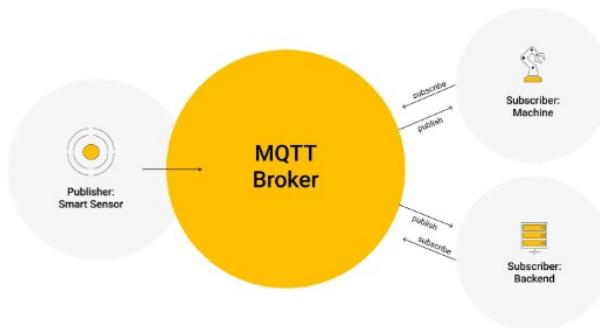
Bên cạnh các ưu điểm, SQLite cũng tồn tại một số hạn chế cần lưu ý. Do đặc điểm kiến trúc nhúng và cơ chế khóa khi ghi dữ liệu, SQLite không phù hợp cho các hệ thống có số lượng người dùng lớn hoặc yêu cầu truy cập ghi đồng thời ở mức cao. Ngoài ra, SQLite không được tối ưu cho việc xử lý dữ liệu có dung lượng lớn hoặc các hệ thống có tải truy cập cao liên tục. So với các hệ quản trị cơ sở dữ liệu máy chủ, SQLite cũng thiếu một số tính năng nâng cao như cơ chế phân quyền phức tạp, quản lý người dùng riêng biệt và các công cụ quản trị chuyên sâu. Vì vậy, SQLite thường được sử dụng như một giải pháp lưu trữ cục bộ hoặc trung gian, thay vì làm cơ sở dữ liệu trung tâm cho các hệ thống quy mô lớn.

## 2.7 TỔNG QUAN VỀ GIAO THỨC MQTT VÀ MOSQUITTO

### 2.7.1 Giới thiệu chung về giao thức MQTT

MQTT là một giao thức truyền thông nhẹ, được thiết kế dành cho các hệ thống có tài nguyên hạn chế và môi trường mạng không ổn định. Giao thức này hoạt động trên nền TCP/IP, cho phép truyền dữ liệu với độ trễ thấp và chi phí băng thông nhỏ. Nhờ các đặc điểm đó, MQTT được sử dụng rộng rãi trong các hệ thống Internet of Things (IoT) [19].

### 2.7.2 Mô hình truyền thông Publish/Subscribe của MQTT



Hình 2.4: Mô hình Publish/Subscribe của MQTT [20]

MQTT sử dụng mô hình truyền thông Publish/Subscribe, trong đó các thiết bị không giao tiếp trực tiếp với nhau mà thông qua một thành phần trung gian gọi là broker. Thiết bị gửi dữ liệu (publisher) sẽ xuất bản thông tin lên các chủ đề (topic), trong khi thiết bị nhận dữ liệu (subscriber) đăng ký nhận dữ liệu từ các chủ đề tương ứng. Mô hình này giúp giảm sự phụ thuộc giữa các thiết bị, tăng tính linh hoạt và khả năng mở rộng cho hệ thống IoT [20].

### 2.7.3 Giới thiệu về MQTT Broker Mosquitto

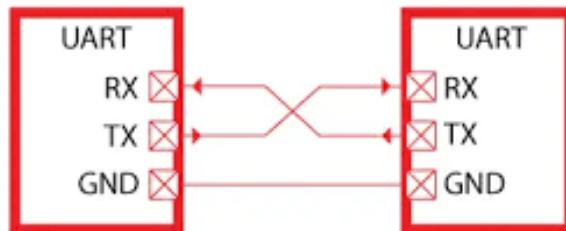
Mosquitto là một MQTT broker mã nguồn mở, tuân thủ đầy đủ các chuẩn của giao thức MQTT. Phần mềm này có kích thước nhỏ, dễ cài đặt và vận hành trên các

nền tảng như Linux và Raspberry Pi. Nhờ tính ổn định và hiệu suất cao, Mosquitto thường được sử dụng trong các hệ thống IoT quy mô nhỏ và vừa, đóng vai trò trung tâm trong việc tiếp nhận, phân phối dữ liệu và lệnh điều khiển giữa các thành phần của hệ thống [21].

## 2.8 TỔNG QUAN VỀ GIAO THỨC UART

### 2.8.1 Nguyên lý hoạt động của UART

UART truyền dữ liệu theo từng khung, bao gồm bit bắt đầu (start bit), các bit dữ liệu, bit chẵn lẻ (tùy chọn) và bit kết thúc (stop bit). Hai thiết bị giao tiếp với nhau thông qua hai đường tín hiệu chính là TX (truyền) và RX (nhận), với các thông số cấu hình thông nhất như tốc độ baud, số bit dữ liệu và số bit dừng để đảm bảo truyền thông chính xác [22]. Dưới đây là hình ảnh minh họa kết nối hai thiết bị theo chuẩn UART:



Hình 2.5: Giao tiếp giữa hai thiết bị qua UART

### 2.8.2 Ưu điểm của UART

UART có ưu điểm là cấu trúc đơn giản, dễ cấu hình và không yêu cầu nhiều chân kết nối. Giao thức này phù hợp cho các kết nối điểm–điểm trong khoảng cách ngắn, có độ ổn định cao và được hỗ trợ sẵn trên hầu hết các vi điều khiển hiện nay.

### **2.8.3 Nhược điểm của UART**

Do hoạt động theo mô hình điểm–điểm, UART không phù hợp cho các hệ thống có nhiều thiết bị cùng giao tiếp trên một bus. Ngoài ra, tốc độ truyền của UART thấp hơn so với một số giao thức nối tiếp khác như SPI, và dễ bị ảnh hưởng bởi nhiễu khi truyền trên khoảng cách xa.

## **2.9 TỔNG QUAN VỀ RASPBERRY PI 4**

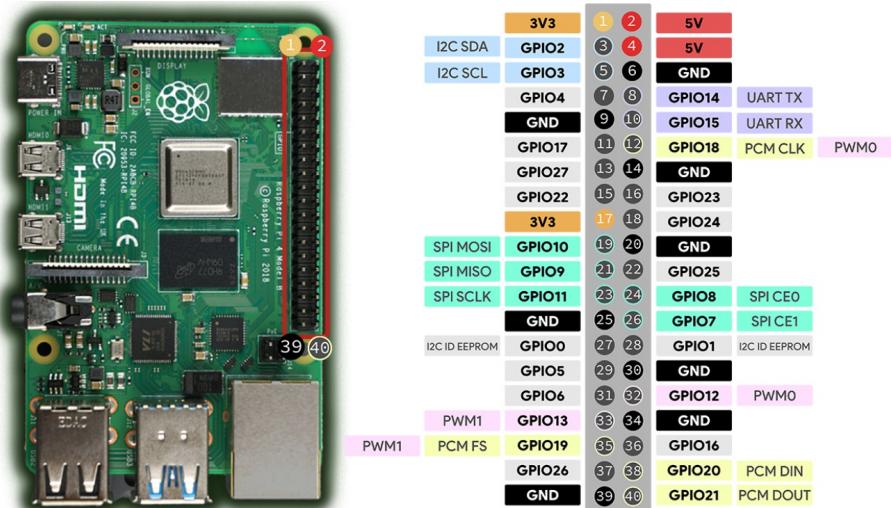
Raspberry Pi 4 là một máy tính nhúng bo mạch đơn, tích hợp đầy đủ các thành phần cơ bản của một hệ thống máy tính hoàn chỉnh. Thiết bị sử dụng bộ vi xử lý ARM đa nhân, bộ nhớ RAM, các cổng giao tiếp ngoại vi và khả năng kết nối mạng. Trên bo mạch có các cổng USB, HDMI, Ethernet, GPIO và các giao tiếp chuẩn như UART, SPI, I2C, cho phép Raspberry Pi 4 dễ dàng kết nối với các thiết bị ngoại vi và hệ thống nhúng khác [23].

### **2.9.1 Cấu tạo của máy tính Raspberry Pi 4**

Raspberry Pi 4 được thiết kế để cung cấp một nền tảng tính toán hoàn chỉnh với đầy đủ các thành phần cơ bản của một hệ thống máy tính. Raspberry Pi 4 tích hợp bộ xử lý trung tâm hiệu năng cao, bộ nhớ, các khối giao tiếp ngoại vi và các cổng kết nối tiêu chuẩn, cho phép vận hành hệ điều hành Linux và triển khai các ứng dụng nhúng phức tạp trong lĩnh vực IoT và điều khiển công nghiệp. Khác với vi điều khiển thông thường, Raspberry Pi 4 sử dụng kiến trúc SoC Broadcom BCM2711, cho phép thực thi đa nhiệm, quản lý tiến trình, bộ nhớ và mạng [23].

Các thành phần chính cấu tạo nên Raspberry Pi 4 bao gồm:

- Bộ xử lý trung tâm (CPU)
- Bộ nhớ
- Các cổng giao tiếp ngoại vi
- Hệ thống cấp nguồn và quản lý năng lượng



Hình 2.6: Sơ đồ chân của Raspberry Pi 4

Máy tính Raspberry Pi 4 có tất cả 40 chân được gán cho những chức năng khác nhau. Cụ thể bên dưới là bảng trình bày chức năng của các chân.

Bảng 2.1: Chức năng các nhóm chân trên Raspberry Pi 4

Nhóm chân	Chức năng hỗ trợ	Số lượng chân
GPIO	INPUT / OUTPUT	28
I2C	Giao tiếp nối tiếp đồng bộ	2
SPI	Giao tiếp truyền thông nối tiếp	2
UART	Giao tiếp nối tiếp không đồng bộ	2
PWM	Điều chế độ rộng xung	4
PCM/I2S	Giao tiếp âm thanh số	1
3.3V	Chân ra áp 3.3V	2
5V	Chân nguồn	2
GND	Nối đất	8

Tiếp theo, ta sẽ khảo sát các tính năng nổi bật của Raspberry Pi 4:

Bảng 2.2: Các tính năng chính của Raspberry Pi 4

Tính năng	Mô tả
Bộ xử lý	SoC Broadcom BCM2711, CPU ARM Cortex-A72 64-bit, 4 nhân, xung nhịp lên đến 1.5 GHz
Bộ nhớ RAM	Hỗ trợ RAM LPDDR4 với các phiên bản 2GB, 4GB và 8GB
Hệ điều hành	Hỗ trợ Linux (Raspberry Pi OS, Ubuntu, Debian, ...)
Kết nối mạng	Ethernet Gigabit, Wi-Fi chuẩn 802.11ac (2.4 GHz / 5 GHz), Bluetooth 5.0
Giao tiếp ngoại vi	GPIO, UART, I2C, SPI, PWM
Khả năng hiển thị	Hỗ trợ xuất hình ảnh thông qua hai cổng micro-HDMI, độ phân giải lên đến 4K
Lưu trữ	Sử dụng thẻ nhớ microSD, hỗ trợ khởi động từ thiết bị lưu trữ USB

### 2.9.2 Quản lý năng lượng trên Raspberry Pi 4

Raspberry Pi 4 được thiết kế để hoạt động liên tục với mức tiêu thụ năng lượng thấp, đồng thời hỗ trợ nhiều cơ chế quản lý năng lượng nhằm tối ưu hiệu suất và độ ổn định của hệ thống. Việc quản lý năng lượng hiệu quả đặc biệt quan trọng khi Raspberry Pi được sử dụng làm gateway trong các hệ thống IoT vận hành dài hạn.

Máy tính sử dụng nguồn cấp 5V 3A thông qua cổng USB-C, tích hợp các mạch quản lý nguồn để phân phối điện áp ổn định cho CPU, RAM và các ngoại vi. Hệ thống tự động giám sát điện áp đầu vào, cảnh báo và hạn chế hiệu năng khi xảy ra hiện tượng sụt áp nhằm bảo vệ phần cứng.

## 2.10 TỔNG QUAN VỀ CÁC THIẾT BỊ DÙNG TRONG HỆ THỐNG

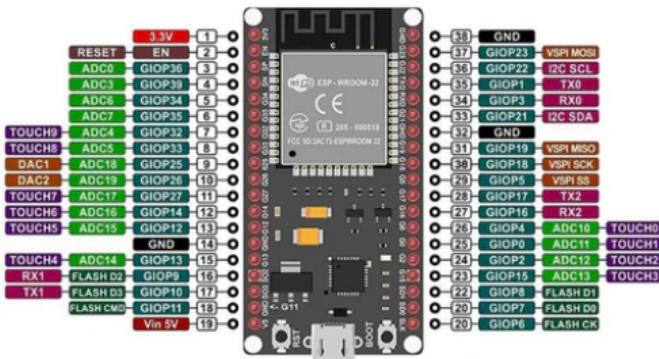
### 2.10.1 Giới thiệu về ESP32 WROOM32 Module

ESP32 WROOM32 Module KIT là một vi điều khiển tích hợp Wi-Fi và Bluetooth do Espressif Systems phát triển, được thiết kế hướng tới các ứng dụng IoT và hệ thống nhúng có kết nối mạng. Module ESP32 tích hợp đầy đủ các khối chức năng cần thiết trên một chip, giúp giảm độ phức tạp phần cứng và tăng tính linh hoạt

trong triển khai hệ thống [24][25].

### 2.10.1.1 Cấu tạo của ESP32 WROOM32 Module KIT

Về cấu trúc phần cứng, ESP32 bao gồm bộ xử lý trung tâm kiến trúc Xtensa 32-bit (dual-core hoặc single-core tùy phiên bản), bộ nhớ RAM và Flash tích hợp, cùng các khối ngoại vi như GPIO, UART, SPI, I2C, ADC, PWM. Ngoài ra, ESP32 còn tích hợp sẵn module Wi-Fi 802.11 b/g/n và Bluetooth (Classic/BLE), cho phép kết nối trực tiếp với mạng và các thiết bị khác mà không cần phần cứng bổ sung [24].



Hình 2.7: Sơ đồ nối chân của ESP32 WROOM32 Module 38 chân

Với module này, có tất cả 38 chân với các nhóm chức năng sau:

Bảng 2.3: Các chức năng các nhóm chân trên module ESP32-WROOM-32

Nhóm chân	Chức năng hỗ trợ	Số lượng
GPIO	INPUT / OUTPUT	25
I2C	Giao tiếp nối tiếp đồng bộ	2
SPI	Giao tiếp truyền thông nối tiếp tốc độ cao	2
UART	Giao tiếp nối tiếp không đồng bộ	3
PWM	Điều chế độ rộng xung	16
ADC	Bộ chuyển đổi tương tự–số độ phân giải 12-bit	18
DAC	Bộ chuyển đổi số–tương tự	2
Touch	Cảm biến cảm ứng điện dung	9–10
5V	Chân cấp nguồn 5V (VIN)	1
3.3V	Chân ra 3.3V	1
GND	Chân nối đất	3
EN	Chân kích hoạt (Enable/Reset chip)	1

Module ESP32 WROOM32 được cấp nguồn 5V vào USB, từ 5V này sẽ đi qua LDO chuyển đổi từ 5V sang 3.3V để cấp cho MCU chính và các ngoại vi trên module.

Bảng 2.4: Các tính năng nổi bật của module ESP32-WROOM-32

Tính năng	Mô tả
Kiến trúc xử lý	Dual-core Xtensa LX6 32-bit, xung nhịp tối đa 240 MHz
Kết nối không dây	Tích hợp Wi-Fi 802.11 b/g/n và Bluetooth Classic/BLE
Bộ nhớ	Flash tích hợp (thường 4 MB), SRAM nội lớn, hỗ trợ PSRAM ngoài
Ngoại vi analog	ADC 12-bit, DAC 8-bit, cảm ứng điện dung Touch
Chuẩn truyền thông	UART, SPI, I2C, I2S, PWM
Quản lý năng lượng	Hỗ trợ nhiều chế độ tiết kiệm năng lượng: Modem Sleep, Light Sleep, Deep Sleep

### 2.10.1.2 Quản lý năng lượng trên Module ESP32

ESP32 được thiết kế hướng tới các ứng dụng IoT yêu cầu tiết kiệm năng lượng, do đó vi điều khiển này hỗ trợ nhiều chế độ quản lý năng lượng khác nhau nhằm tối ưu mức tiêu thụ điện năng trong quá trình vận hành. Tùy theo trạng thái hoạt động của hệ thống và yêu cầu ứng dụng, ESP32 có thể chuyển đổi linh hoạt giữa các chế độ năng lượng để cân bằng giữa hiệu năng xử lý và mức tiêu thụ công suất.

Bảng 2.5: Các chế độ quản lý năng lượng tiêu biểu của ESP32

Thành phần	Active Mode	Light Sleep	Deep Sleep
CPU	Hoạt động	Tạm dừng	Tắt
Wi-Fi / Bluetooth	Hoạt động	Tắt	Tắt
Bộ nhớ RAM	Hoạt động	Giữ dữ liệu	Mất dữ liệu
RTC (Real-Time Clock)	Hoạt động	Hoạt động	Hoạt động
Bộ xử lý ULP	Hoạt động	Hoạt động	Có thể hoạt động
Dòng tiêu thụ	Cao	Thấp	Rất thấp

Trong ba chế độ trên, chế độ Active được sử dụng khi hệ thống cần xử lý dữ liệu và truyền thông liên tục. Chế độ Light Sleep phù hợp cho các giai đoạn chờ ngắn nhằm tiết kiệm năng lượng mà vẫn giữ trạng thái hệ thống. Chế độ Deep Sleep được sử dụng khi hệ thống không cần hoạt động trong thời gian dài, giúp giảm tối đa dòng tiêu thụ năng lượng.

### 2.10.2 Giới thiệu về STM32F103C8T6

STM32F103C8T6 là vi điều khiển 32-bit thuộc dòng STM32F1 được phát triển bởi STMicroelectronics, sử dụng lõi xử lý ARM Cortex-M3. Dòng vi điều khiển này được thiết kế hướng tới các ứng dụng nhúng yêu cầu hiệu năng ổn định, khả năng xử lý thời gian thực và tiêu thụ năng lượng thấp[26].

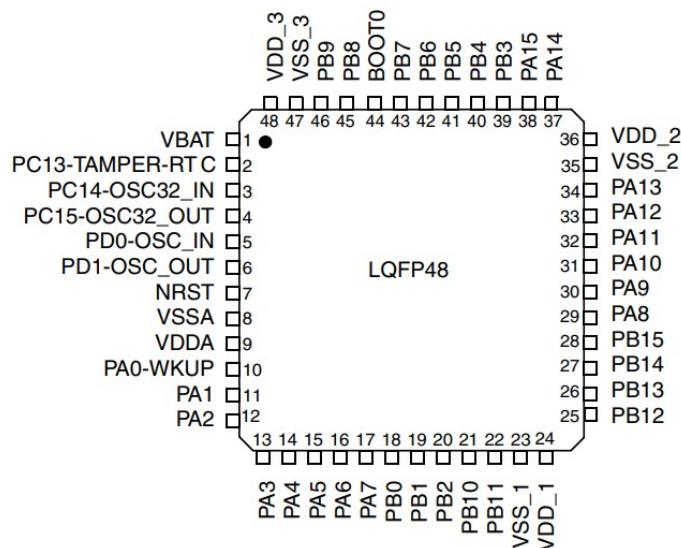
Với xung nhịp hoạt động tối đa 72 MHz, bộ nhớ Flash 64 KB và SRAM 20 KB, STM32F103C8T6 đáp ứng tốt các ứng dụng điều khiển công nghiệp và thu thập

dữ liệu cảm biến. Ngoài ra, vi điều khiển này được tích hợp đa dạng ngoại vi như GPIO, ADC, timer, UART, SPI, I<sup>2</sup>C, giúp thuận tiện trong việc kết nối với các cảm biến và thiết bị chấp hành [26].

### 2.10.2.1 Cấu tạo của STM32F103C8T6

Về mặt cấu tạo, STM32F103C8T6 bao gồm các khối chức năng chính sau:

- Lõi xử lý
- Bộ nhớ
- Khối GPIO
- Khối ADC
- Timer và bộ tạo xung PWM
- Giao tiếp truyền thông



Hình 2.8: Sơ đồ chân của STM32F103C8T6 đóng gói kiểu LQFP48

Với kiểu đóng gói LQFP48, vi điều khiển được chia thành các nhóm chức năng sau:

Bảng 2.6: Chức năng các nhóm chân trên vi điều khiển STM32F103C8T6

Nhóm chân	Chức năng hỗ trợ	Số lượng
GPIO	Chân vào/ra số đa chức năng (Input/Output), hỗ trợ ngắt ngoài	37
ADC	Bộ chuyển đổi tương tự–số độ phân giải 12-bit	10
Timer / PWM	Bộ định thời, tạo xung PWM, đo xung và đếm sự kiện	15
USART / UART	Giao tiếp nối tiếp không đồng bộ (USART1, USART2, USART3)	3
SPI	Giao tiếp truyền thông nối tiếp tốc độ cao	2
I2C	Giao tiếp nối tiếp đồng bộ	2
CAN	Giao tiếp mạng công nghiệp CAN	1
USB	Giao tiếp USB 2.0 Full-Speed (Device)	1
RTC / Backup	Chân hỗ trợ RTC, pin VBAT, vùng nhớ Backup	4
Nguồn (VDD/VDDA)	Chân cấp nguồn cho lõi số và khối analog	4
GND (VSS/VSSA)	Chân nối đất cho lõi số và khối analog	5
Clock / Reset	Chân thạch anh ngoài, reset, boot cấu hình	6

Bảng 2.7: Các tính năng nổi bật của vi điều khiển STM32F103C8T6

Tính năng	Mô tả
Kiến trúc xử lý	ARM Cortex-M3 32-bit, xung nhịp tối đa 72 MHz
Bộ nhớ	Flash 64 KB, SRAM 20 KB, hỗ trợ vùng nhớ Backup
Ngoại vi analog	ADC 12-bit (10 kênh), hỗ trợ đo tín hiệu cảm biến chính xác
Bộ định thời	Nhiều timer 16-bit và 32-bit, hỗ trợ PWM, Input Capture, Output Compare
Chuẩn	USART, SPI, I2C, CAN, USB Full-Speed (Device)
Quản lý năng lượng	Hỗ trợ các chế độ tiết kiệm năng lượng: Sleep, Stop, Standby
Độ ổn định	Hoạt động ổn định trong môi trường công nghiệp, thời gian thực cao

### 2.10.2.2 Quản lý năng lượng trên STM32F103C8T6

STM32F103C8T6 được thiết kế cho các hệ thống nhúng và điều khiển thời gian thực, do đó vi điều khiển này cung cấp nhiều chế độ quản lý năng lượng nhằm giảm mức tiêu thụ điện năng trong các trạng thái hoạt động khác nhau. Việc sử dụng linh hoạt các chế độ năng lượng cho phép hệ thống cân bằng giữa hiệu năng xử lý, độ ổn định và yêu cầu tiết kiệm năng lượng, đặc biệt trong các ứng dụng giám sát và điều khiển liên tục.

Bảng 2.8: Các chế độ quản lý năng lượng tiêu biểu của STM32F103C8T6

Thành phần	Run Mode	Sleep Mode	Stop / Standby Mode
CPU	Hoạt động	Tạm dừng	Tắt
Xung hệ thống	Hoạt động	Hoạt động	Tắt
Bộ nhớ RAM	Hoạt động	Giữ dữ liệu	Giữ (Stop) / Mát (Standby)
Ngoại vi	Hoạt động	Hoạt động chọn lọc	Tắt
RTC	Hoạt động	Hoạt động	Hoạt động
Dòng tiêu thụ	Cao	Trung bình	Rất thấp

Trong các chế độ trên, Run Mode được sử dụng khi hệ thống cần xử lý và điều khiển liên tục với hiệu năng cao. Sleep Mode cho phép tạm dừng CPU trong khi vẫn duy trì hoạt động của các ngoại vi cần thiết, phù hợp với các giai đoạn chờ ngắn giữa các tác vụ. Stop Mode và Standby Mode được sử dụng khi hệ thống không cần hoạt động trong thời gian dài, giúp giảm đáng kể mức tiêu thụ năng lượng, đặc biệt phù hợp với các hệ thống giám sát hoạt động theo chu kỳ.

### 2.10.3 Giới thiệu về cảm biến nhiệt độ DS18B20

Cảm biến nhiệt độ DS18B20 là cảm biến đo nhiệt độ kỹ thuật số được sử dụng rộng rãi trong các hệ thống nhúng và ứng dụng IoT nhờ độ chính xác cao, giao tiếp đơn giản và khả năng hoạt động ổn định trong môi trường công nghiệp. Cảm biến sử dụng giao thức truyền thông 1-Wire [27].



Hình 2.9: Cảm biến DS18B20

Thông số kỹ thuật của cảm biến:

Bảng 2.9: Thông số kỹ thuật của cảm biến DS18B20

Thông số	Giá trị
Điện áp hoạt động	3.0V – 5.5V
Dải đo nhiệt độ	-55°C đến +125°C
Độ chính xác	±0.5°C
Độ phân giải	9 – 12 bit
Giao tiếp	1-Wire
Thời gian chuyển đổi	Tối đa 750 ms (12-bit)
Dòng tiêu thụ	khoảng 1mA
Nhiệt độ hoạt động	-55°C đến +125°C

Sơ đồ chân: Cảm biến DS18B20 có 3 chân kết nối chính, phù hợp cho việc tích hợp vào các hệ thống nhúng có tài nguyên phần cứng hạn chế.

Bảng 2.10: Chức năng các chân của cảm biến DS18B20

Chân	Chức năng
VDD	Cấp nguồn
DQ	Chân dữ liệu 1-Wire
GND	Nối đất

#### 2.10.4 Giới thiệu về cảm biến PH

Cảm biến Gravity: Analog pH Meter V1.1 của DFRobot là một trong những module đo pH phổ biến, được thiết kế dành cho các ứng dụng IoT và các hệ thống giám sát chất lượng nước. Module này cung cấp một giải pháp hoàn chỉnh nhằm chuyển đổi tín hiệu điện hóa rất nhỏ từ điện cực pH thành tín hiệu điện áp analog ổn định, phù hợp để vi điều khiển đọc và xử lý [28].



Hình 2.10: Cảm biến PH

Một bộ cảm biến gồm 2 thành phần chính:

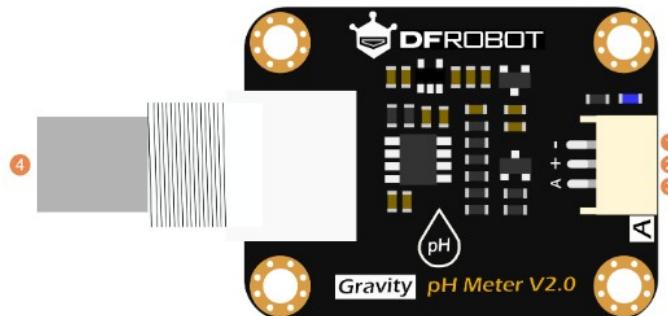
- Điện cực pH: là điện cực dạng thủy tinh sử dụng nguyên lý điện hóa, bên trong có chứa dung dịch đệm và màng thấm thấu ion.
- Board xử lý tín hiệu: có nhiệm vụ khuếch đại và dịch mức tín hiệu điện áp nhỏ (đơn vị mV) từ điện cực pH lên mức điện áp analog chuẩn.

Thông số kỹ thuật quan trọng của cảm biến:

Bảng 2.11: Thông số kỹ thuật cảm biến Gravity: Analog pH Meter V1.1

Thông số	Giá trị
Dải đo	0 – 14 pH
Điện áp cấp nguồn	3.3V - 5.0 V DC
Dòng tiêu thụ	Khoảng 5 – 10 mA
Ngõ ra tín hiệu	Analog (0 – 5 V)
Nhiệt độ làm việc	0 – 60 °C

Sơ đồ chân của board xử lý tín hiệu:



Hình 2.11: Sơ đồ chân board xử lý tín hiệu cảm biến PH

Bảng 2.12: Chân và chức năng của các chân trên cảm biến PH

Chân	Ký hiệu	Chức năng
1	-	Nối đất
2	+	Chân nguồn
3	A	Chân tín hiệu
4		Điện cực pH

## 2.10.5 Giới thiệu về cảm biến độ đục

Cảm biến độ đục Gravity: Analog Turbidity Sensor (SKU: SEN0189) của DFRobot là một module cảm biến phổ biến được sử dụng trong các hệ thống giám sát chất lượng nước, đặc biệt trong lĩnh vực xử lý nước thải, nuôi trồng thủy sản và các ứng dụng IoT môi trường. Cảm biến cho phép đánh giá mức độ trong – đục của nước

through the process of measuring the intensity of light that is scattered by particles in the water [29].



Hình 2.12: Cảm biến độ đục SEN0189

Bộ cảm biến độ đục SEN0189 được cấu tạo gồm hai thành phần chính:

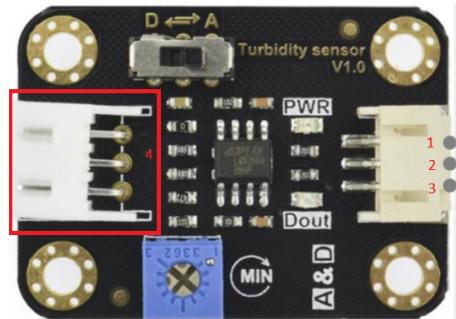
- Đầu dò: được tích hợp một đèn LED hồng ngoại và photodiode thu tín hiệu. Khi ánh sáng qua nước, các hạt lơ lửng sẽ làm thay đổi cường độ ánh sáng thu được, từ đó phản ánh mức độ đục của nước.
- Mạch xử lý tín hiệu: có nhiệm vụ khuếch đại và xử lý tín hiệu analog từ đầu dò, đồng thời chuyển đổi tín hiệu quang học thành điện áp đầu ra trong khoảng cho phép để vi điều khiển có thể đọc và xử lý.

Thông số kỹ thuật cần chú ý của cảm biến:

Bảng 2.13: Thông số kỹ thuật cảm biến độ đục Gravity: Analog Turbidity Sensor

Thông số	Giá trị
Dải đo độ đục	0 – 1000 NTU
Điện áp cấp nguồn	3.3V – 5.0 V DC
Dòng tiêu thụ	Khoảng 30 – 40 mA
Ngõ ra tín hiệu	Analog (0 – 4.5 V)
Nhiệt độ làm việc	5 – 90 °C
Thời gian đáp ứng	< 500 ms
Chuẩn kết nối	Giao tiếp analog

Sơ đồ chân của mạch xử lý tín hiệu:



Hình 2.13: Sơ đồ chân của cảm biến độ đục SEN0189

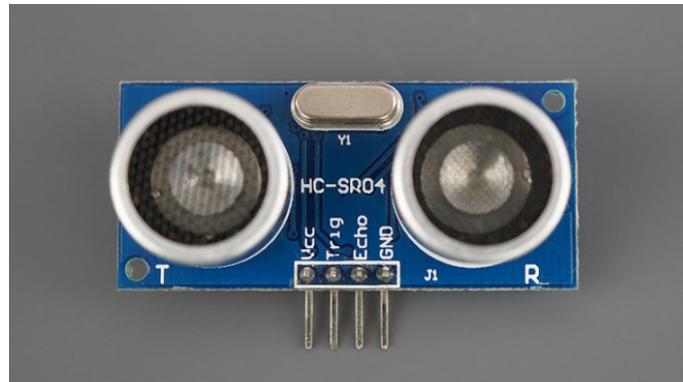
Bảng 2.14: Chân và chức năng của các chân trên cảm biến độ đục SEN0189

Chân	Chức năng
1	Nối đất
2	Chân nguồn
3	Chân tín hiệu
4	Đầu dò

## 2.10.6 Giới thiệu về cảm biến khoảng cách HC-SR04

Cảm biến HC-SR04 là cảm biến đo khoảng cách không tiếp xúc sử dụng sóng siêu âm, được ứng dụng rộng rãi trong các hệ thống đo mức, robot tự hành và các

mô hình giám sát – điều khiển công nghiệp quy mô nhỏ [30].



Hình 2.14: Cảm biến khoảng cách HC-SR04

Thông số kỹ thuật của cảm biến:

Bảng 2.15: Thông số kỹ thuật cảm biến khoảng cách siêu âm HC-SR04

Thông số	Giá trị
Nguyên lý đo	Sóng siêu âm
Dải đo	2 – 400 cm
Tần số sóng siêu âm	40 kHz
Điện áp cấp nguồn	5 V DC
Dòng tiêu thụ	Khoảng 15 mA
Nhiệt độ làm việc	0 – 70 °C

Sơ đồ chân của cảm biến:

Bảng 2.16: Chân và chức năng của các chân trên cảm biến HC-SR04

Ký hiệu	Chức năng
Vcc	Chân nguồn
Trig	Ngõ vào kích
Echo	Ngõ ra tín hiệu
GND	Nối đất

Sơ bộ về nguyên lý hoạt động của cảm biến: cảm biến khoảng cách siêu âm

HC-SR04 hoạt động dựa trên nguyên lý đo thời gian truyền của sóng siêu âm trong không khí. Khi chân TRIG được kích bằng một xung tối thiểu là 10  $\mu$ s, cảm biến phát ra sóng siêu âm có tần số 40 kHz. Sóng này truyền đi, phản xạ khi gặp vật cản và quay trở lại bộ thu của cảm biến. Trong suốt quá trình này, chân ECHO duy trì mức logic cao; thời gian mức cao của tín hiệu ECHO tương ứng với thời gian sóng siêu âm đi và về. Từ thời gian đó ta có được khoảng cách tính theo công thức.

$$\text{Khoảng cách (cm)} = (\text{Thời gian xung ở mức cao } (\mu\text{s}) \times 0.0343) / 2$$

Với 0.0343 là tốc độ âm thanh đã được đổi về đơn vị cm/ $\mu$ s (tốc độ gốc là 343 m/s).

# CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

## 3.1 YÊU CẦU HỆ THỐNG

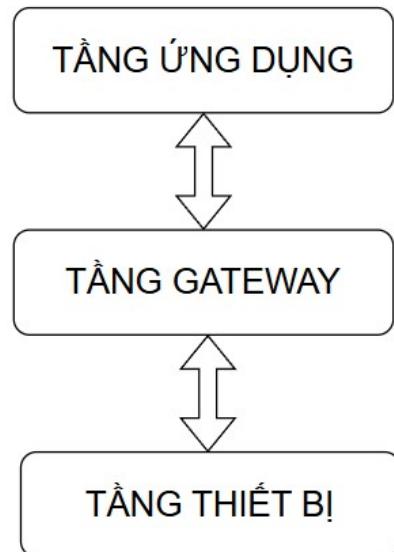
Xuất phát từ những yêu cầu đặt ra ban đầu của đề tài THIẾT KẾ VÀ THI CÔNG MÔ HÌNH IOT BẢO MẬT ĐA TẦNG ỨNG DỤNG TRONG QUY TRÌNH XỬ LÝ NƯỚC THẢI CÔNG NGHIỆP, tôi tiến hành xây dựng dự án đáp ứng đầy đủ các tính năng sau:

- Đảm bảo thực hiện việc bảo mật dữ liệu ở nhiều tầng.
- Tại gateway là Raspberry Pi phải thực hiện kiểm soát, phân phối và lọc lệnh điều khiển tại lớp kernel, chỉ cho phép các kết nối và lệnh sử dụng các cổng mạng hợp lệ.
- Thu thập và giám sát liên tục các thông số cảm biến được lựa chọn.
- Truyền thông dữ liệu ổn định giữa các tầng.
- Hỗ trợ điều khiển các cơ cấu chấp hành từ xa thông qua giao diện web.
- Phân quyền người dùng có thể điều khiển theo vai trò.
- Hiển thị dữ liệu giám sát và trạng thái thiết bị trên giao diện web một cách trực quan.
- Các cảm biến được đọc theo thời gian thực.
- Tối ưu độ trễ của hệ thống xuống thấp nhất có thể.

## 3.2 KIẾN TRÚC ĐỀ XUẤT

Mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp được thiết kế theo kiến trúc phân tầng, nhằm đảm bảo tính rõ ràng trong tổ chức chức năng, dễ mở rộng và phù hợp với môi trường công nghiệp có yêu cầu cao về an toàn và bảo mật. Kiến trúc tổng thể của hệ thống bao gồm nhiều tầng

chức năng, phối hợp với nhau để thực hiện các nhiệm vụ giám sát, điều khiển và quản lý hệ thống.



Hình 3.1: Kiến trúc phân tầng của hệ thống

Kiến trúc bao gồm tầng ứng dụng, tầng gateway, tầng thiết bị. Với tầng thiết bị sẽ chứa các thành phần: khói vi điều khiển, khói cảm biến, khói điều khiển cơ cấu chấp hành, khói nguồn thiết bị. Tầng gateway sẽ bao gồm khói gateway và khói nguồn gateway. Cuối cùng, tầng ứng dụng sẽ quản lý khói lưu trữ, khói hiển thị và điều khiển. Các khói được đề cập, sẽ được nói rõ hơn ở phần sơ đồ khói.

Luồng hoạt động của hệ thống được tổ chức theo hai hướng chính. Ở luồng giám sát, dữ liệu cảm biến được thu thập từ tầng thiết bị, truyền qua tầng trung gian và hiển thị trên giao diện web theo thời gian thực thuộc tầng ứng dụng. Ở luồng điều khiển, các lệnh điều khiển được người dùng gửi từ giao diện web, kiểm tra quyền truy cập và tính hợp lệ tại tầng gateway, sau đó chuyển tiếp xuống tầng thiết bị. Kiến trúc này cho phép hệ thống vừa đáp ứng yêu cầu giám sát – điều khiển linh hoạt, vừa đảm bảo phân quyền và kiểm soát truy cập trong toàn bộ quá trình vận hành.

### **3.3 PHƯƠNG ÁN GIÁM SÁT CHO MÔ HÌNH**

Trên cơ sở khảo sát các quy trình xử lý nước thải công nghiệp điển hình và các thông số cần giám sát trong thực tế đã trình bày ở chương trước, chương này tập trung xây dựng phương án giám sát và điều khiển cho mô hình IoT được triển khai trong đề tài. Phương án giám sát được thiết kế theo hướng đại diện, lựa chọn các công đoạn và thông số tiêu biểu nhằm phản ánh trạng thái vận hành của hệ thống xử lý nước thải, đồng thời phù hợp với quy mô mô phỏng và mục tiêu nghiên cứu của đề tài.

#### **3.3.1 Lựa chọn bể giám sát trong mô hình**

Trong thực tế, một hệ thống xử lý nước thải công nghiệp có thể bao gồm nhiều bể và công đoạn khác nhau như bể thu gom, bể điều hòa, bể xử lý sinh học, bể lắng và bể xả thải. Tuy nhiên, trong phạm vi mô hình nghiên cứu, đề tài tập trung giám sát hai bể đại diện, bao gồm:

- Bể thu gom (Collection Tank): là nơi tiếp nhận nước thải đầu vào từ quá trình sản xuất. Việc giám sát bể này giúp đánh giá tình trạng nước thải ban đầu, kiểm soát mức nước, phát hiện sớm các bất thường và làm cơ sở cho các quyết định điều khiển ở các công đoạn phía sau.
- Bể nước thải sau xử lý (Discharge Tank): đại diện cho đầu ra của hệ thống xử lý. Các thông số tại bể này phản ánh hiệu quả xử lý tổng thể và là căn cứ để đánh giá mức độ đáp ứng yêu cầu xả thải theo quy chuẩn môi trường.

Việc lựa chọn hai bể trên cho phép mô hình vừa phản ánh được trạng thái đầu vào – đầu ra của hệ thống, vừa đảm bảo tính đơn giản, phù hợp cho mục đích mô phỏng và thử nghiệm giải pháp giám sát – điều khiển bằng IoT.

#### **3.3.2 Lựa chọn các thông số cần giám sát**

Dựa trên các thông số giám sát phổ biến trong thực tế vận hành hệ thống xử lý nước thải công nghiệp, đồng thời xét đến khả năng triển khai trên mô hình, đề tài

lựa chọn các thông số giám sát chính sau:

- Nhiệt độ nước: phản ánh điều kiện môi trường ảnh hưởng đến quá trình xử lý sinh học và hoạt động của các thiết bị trong hệ thống.
- Giá trị pH: là thông số quan trọng để đánh giá tính axit – kiềm của nước thải, ảnh hưởng trực tiếp đến hiệu suất xử lý và an toàn thiết bị.
- Độ đục: đại diện cho hàm lượng chất rắn lơ lửng trong nước, giúp đánh giá sơ bộ chất lượng nước và hiệu quả quá trình lắng – lọc.
- Mực nước: sử dụng để giám sát trạng thái bể, phát hiện tình trạng đầy hoặc cạn, đồng thời làm cơ sở cho các thuật toán điều khiển.

Để phục vụ cho việc giám sát và cảnh báo trạng thái vận hành của hệ thống, mỗi thông số được xây dựng các ngưỡng giám sát với ba mức gồm: an toàn, cảnh báo và cảnh báo nghiêm trọng. Các ngưỡng cụ thể được lựa chọn như sau:

- Nhiệt độ nước:
  - Ngưỡng an toàn: từ 20 °C đến 35 °C.
  - Ngưỡng cảnh báo: từ 15 °C đến dưới 20 °C hoặc trên 35 °C đến 40 °C.
  - Ngưỡng cảnh báo nghiêm trọng: nhỏ hơn 15 °C hoặc lớn hơn 40 °C.
- Giá trị pH:
  - Ngưỡng an toàn: từ 6.5 đến 8.5.
  - Ngưỡng cảnh báo: từ 5.5 đến dưới 6.5 hoặc trên 8.5 đến 9.0.
  - Ngưỡng cảnh báo nghiêm trọng: nhỏ hơn 5.5 hoặc lớn hơn 9.0.
- Độ đục:
  - Ngưỡng an toàn: nhỏ hơn 100 NTU.
  - Ngưỡng cảnh báo: từ 100 NTU đến 250 NTU.

- Ngưỡng cảnh báo nghiêm trọng: lớn hơn 250 NTU.
- Mực nước:
  - Ngưỡng an toàn: từ 30% đến 80% chiều cao bể.
  - Ngưỡng cảnh báo: từ 20% đến dưới 30% hoặc trên 80% đến 90% chiều cao bể.
  - Ngưỡng cảnh báo nghiêm trọng: nhỏ hơn 20% hoặc lớn hơn 90% chiều cao bể.

Các ngưỡng giám sát được xây dựng nhằm phục vụ cho mục tiêu nghiên cứu và mô phỏng hệ thống, có thể được hiệu chỉnh linh hoạt khi triển khai trong các điều kiện vận hành thực tế khác nhau.

### **3.3.3 Lựa chọn thiết bị điều khiển trong mô hình**

Bên cạnh chức năng giám sát, mô hình còn hỗ trợ điều khiển một số thiết bị nhằm mô phỏng hoạt động của hệ thống xử lý nước thải trong thực tế. Các thiết bị điều khiển trong mô hình bao gồm:

- Bơm nước: mô phỏng quá trình bơm nước giữa các bể.
- Van điện từ: điều khiển dòng chảy của nước.
- Động cơ khuấy: hỗ trợ quá trình trộn và vận hành hệ thống.
- Hệ thống chiếu sáng: mô phỏng quá trình chiếu sáng của bộ phận.

Trong phạm vi đồ án, các thiết bị trên được mô phỏng bằng module relay, cho phép kiểm chứng nguyên lý điều khiển và tích hợp với hệ thống IoT mà không cần triển khai tải công suất lớn.

### **3.3.4 Định hướng tích hợp giám sát và điều khiển trong mô hình**

Phương án giám sát và điều khiển trong mô hình được xây dựng theo hướng gán cụ thể các thiết bị chấp hành và cảm biến cho từng bể xử lý, nhằm phản ánh sát

nhất cách tổ chức và vận hành của các hệ thống xử lý nước thải trong thực tế. Mỗi bể trong mô hình được xem như một đơn vị vận hành độc lập, bao gồm tập hợp các thiết bị và thông số giám sát tương ứng.

Cụ thể, bể thu gom được tích hợp các thiết bị và cảm biến sau:

- Thiết bị chấp hành: đèn chiếu sáng, bơm nước, van điện và quạt khuấy.
- Thông số cần giám sát: nhiệt độ nước, độ đục và mực nước.

Việc giám sát các thông số tại bể thu gom giúp đánh giá tình trạng nước thải đầu vào, đồng thời làm cơ sở cho việc điều khiển bơm, van và quạt khuấy nhằm đảm bảo quá trình vận hành ổn định.

Trong mô hình đề tài, bể thu gom được xem là khu vực tiếp nhận nước thải đầu vào từ các nguồn sản xuất khác nhau trước khi đưa vào các công đoạn xử lý tiếp theo. Tại giai đoạn này, nước thải thường có tính chất biến động mạnh, bao gồm sự dao động về lưu lượng, hàm lượng chất rắn lơ lửng và nhiệt độ, tùy thuộc vào thời điểm xả thải và đặc điểm của quá trình sản xuất.

Đối với giá trị pH, mặc dù là một thông số quan trọng trong đánh giá chất lượng nước thải, nhưng tại bể thu gom, pH chưa mang nhiều ý nghĩa kiểm soát vận hành trực tiếp. Nguyên nhân là do:

- Nước thải tại bể thu gom thường là hỗn hợp của nhiều dòng thải khác nhau, khiến giá trị pH có thể dao động lớn và chưa phản ánh trạng thái ổn định của quá trình xử lý.
- Trong các hệ thống xử lý nước thải thực tế, pH thường được điều chỉnh hoặc ổn định ở các công đoạn xử lý tiếp theo (bể điều hòa, bể sinh học), thay vì kiểm soát chặt ngay tại bể thu gom.

Do đó, trong phạm vi mô hình, đề tài không lựa chọn cảm biến pH tại bể thu gom, mà tập trung giám sát các thông số có ảnh hưởng trực tiếp đến vận hành và an toàn hệ thống.

Đối với bể xả (bể đầu ra), hệ thống được tích hợp:

- Thiết bị chấp hành: đèn chiếu sáng, bơm nước và van điện từ.
- Thông số giám sát: nhiệt độ nước, độ đục, mực nước và giá trị pH.

Các thông số tại bể xả phản ánh trực tiếp chất lượng nước sau xử lý, trong đó pH và độ đục là những chỉ tiêu quan trọng để đánh giá hiệu quả xử lý và khả năng đáp ứng yêu cầu xả thải.

### **3.4 ĐẶC TẢ KỸ THUẬT**

Mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp được thiết kế nhằm phục vụ mục tiêu tích hợp cơ chế phân quyền và bảo mật đa tầng nhằm đảm bảo an toàn khi triển khai trong môi trường công nghiệp có kết nối mạng. Đồng thời, theo dõi trạng thái vận hành, thu thập dữ liệu môi trường và hỗ trợ điều khiển các thiết bị xử lý theo thời gian thực. Để đảm bảo hệ thống hoạt động ổn định và hiệu quả trong thực tế, các yêu cầu kỹ thuật về đầu vào, đầu ra và điều kiện vận hành được đặc tả như sau.

#### **3.4.1 Đầu vào của hệ thống**

##### **Nguồn cấp điện**

Hệ thống được cấp nguồn cho các node thiết bị và gateway thông qua các nguồn DC độc lập phù hợp với từng khối chức năng. Các mạch ổn áp được sử dụng để cung cấp mức điện áp phù hợp cho vi điều khiển, module truyền thông và các ngoại vi.

- Điện áp cấp cho node cảm biến (STM32, ESP32): 3.3V DC
- Điện áp cấp cho gateway Raspberry Pi 4: 5V DC - 3A
- Công suất tiêu thụ toàn hệ thống: nhỏ hơn 10W
- Yêu cầu bảo vệ: chống quá áp, lọc nhiễu nguồn, bảo vệ ngắn mạch và cách ly hợp lý.

### **Tín hiệu cảm biến**

Hệ thống thu thập dữ liệu từ các cảm biến môi trường và quá trình xử lý nước thải được lắp đặt tại các vị trí giám sát.

- Cảm biến pH: Dải đo 0–14 pH, tín hiệu ngõ ra analog.
- Cảm biến độ đục: Dải đo 0–1000 NTU, tín hiệu ngõ ra analog.
- Cảm biến nhiệt độ: Dải đo từ –55 °C đến +125 °C, giao tiếp 1-Wire.
- Cảm biến mực nước: Đo bằng phương pháp siêu âm, dải đo 2–400 cm.

### **Tín hiệu điều khiển từ người dùng**

Hệ thống nhận lệnh điều khiển từ người dùng thông qua giao diện web.

- Hình thức điều khiển: thông qua Web Dashboard.
- Yêu cầu xác thực: đăng nhập bằng tài khoản và mật khẩu
- Quyền điều khiển phụ thuộc vào vai trò người dùng (Admin, User1, User2)

#### **3.4.2 Đầu ra của hệ thống**

##### **Điều khiển thiết bị chấp hành**

Dựa trên lệnh điều khiển từ hệ thống hoặc thuật toán điều khiển tự động, vi điều khiển STM32F103C8T6 thực hiện điều khiển các thiết bị chấp hành trong hệ thống xử lý nước thải. Trong phạm vi đồ án, các thiết bị chấp hành được mô phỏng bằng các module relay, đại diện cho các tải công nghiệp như bơm, van điện tử và quạt khuấy.

Thiết bị điều khiển (mô phỏng):

- Đèn trong bộ phận (Mô phỏng bằng Relay).
- Bơm nước (Mô phỏng bằng Relay).
- Van điện (Mô phỏng bằng Relay).

- Động cơ khuỷu (Mô phỏng bằng Relay).

Hình thức điều khiển:

- Điều khiển bật/tắt thiết bị.
- Điều khiển tự động dựa trên ngưỡng mực nước đã được cấu hình.

Tín hiệu điều khiển:

- Tín hiệu số từ STM32 thông qua module relay 5 V.

Việc sử dụng relay trong mô hình giúp mô phỏng nguyên lý điều khiển thiết bị chấp hành trong hệ thống xử lý nước thải thực tế, đồng thời đảm bảo an toàn và phù hợp với quy mô đồ án.

### **Truyền và hiển thị dữ liệu**

Dữ liệu cảm biến và trạng thái thiết bị được truyền từ STM32 lên gateway Raspberry Pi 4 thông qua module ESP32, sử dụng giao thức MQTT. Các dữ liệu này được hiển thị trên giao diện web nhằm phục vụ giám sát và điều khiển theo thời gian thực.

Giao thức truyền thông:

- STM32 với ESP32: UART.
- ESP32 với Raspberry Pi 4: MQTT.

Chu kỳ cập nhật dữ liệu:

- Thời gian: khoảng 1s.

Dữ liệu hiển thị trên giao diện web:

- Giá trị đọc được từ các cảm biến theo thời gian thực.
- Trạng thái bật/tắt của các relay.
- Cảnh báo khi các thông số vượt ngưỡng cho phép.
- Biểu đồ trực quan hóa dữ liệu đo được.

### **3.4.3 Điều kiện vận hành**

Hệ thống được thiết kế để hoạt động ổn định trong môi trường công nghiệp mô phỏng, phù hợp với điều kiện triển khai thực tế của mô hình xử lý nước thải.

#### **Điều kiện môi trường**

Nhiệt độ làm việc:

- Node cảm biến (gồm STM32, ESP32 và các cảm biến): 0 - 50 °C
- Gateway Raspberry Pi 4: 0 - 50 °C

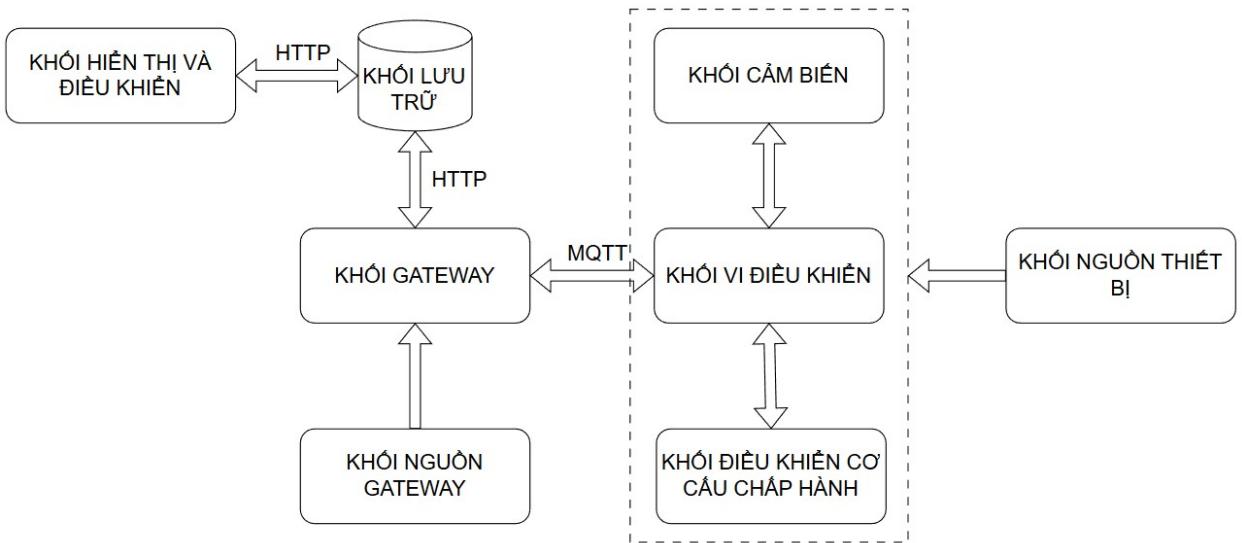
Yêu cầu vận hành:

- Thời gian phản hồi của hệ thống: khoảng 500ms
- Tần suất cập nhật dữ liệu: khoảng 1 Hz (tương ứng chu kỳ cập nhật 1 giây)

Hệ thống hỗ trợ hai chế độ vận hành:

- Chế độ tự động: hệ thống tự động điều khiển relay dựa trên các ngưỡng mực nước và điều kiện đã được cấu hình trước, nhằm đảm bảo vận hành ổn định và an toàn.
- Chế độ thủ công: người dùng có quyền phù hợp có thể trực tiếp điều khiển relay thông qua giao diện web để phục vụ nhu cầu vận hành và kiểm tra hệ thống.

### 3.5 SƠ ĐỒ KHỐI HỆ THỐNG



Hình 3.2: Sơ đồ khái niệm của hệ thống

Chức năng của các khái niệm cụ thể trong hệ thống được mô tả như sau:

- Khối cảm biến: có nhiệm vụ thu thập các thông số đặc trưng của môi trường nước tại hiện trường. Các cảm biến được bố trí tại những vị trí giám sát quan trọng nhằm đo các đại lượng như pH, độ đục, nhiệt độ nước và mực nước. Dữ liệu đo được từ các cảm biến được cung cấp cho khối vi điều khiển để xử lý và truyền đi phục vụ công tác giám sát và điều khiển hệ thống.
- Khối điều khiển cơ cấu chấp hành: có nhiệm vụ thực thi các lệnh điều khiển do khối vi điều khiển phát ra. Trong phạm vi mô hình, các cơ cấu chấp hành như bơm, van điện tử và động cơ khuấy được mô phỏng thông qua các module relay. Khối này đại diện cho các thiết bị công nghiệp trong hệ thống xử lý nước thải thực tế, cho phép kiểm tra và đánh giá nguyên lý điều khiển mà không cần triển khai tải công suất lớn.
- Khối vi điều khiển: là khái niệm đóng vai trò trung tâm xử lý tại tầng thiết bị, thực hiện việc đọc dữ liệu từ các cảm biến, xử lý logic điều khiển và giao tiếp với các khái niệm khác trong mô hình. Vi điều khiển STM32 chịu trách nhiệm điều

khiến các cơ cấu chấp hành thông qua tín hiệu điều khiển số, đồng thời ESP32 sẽ đóng gói và truyền dữ liệu cảm biến lên khói gateway thông qua giao thức truyền thông MQTT. Khối này đảm bảo hệ thống hoạt động ổn định và đáp ứng yêu cầu điều khiển theo thời gian thực ở mức thiết bị.

- Khối gateway: đóng vai trò là trung gian giữa tầng thiết bị và tầng ứng dụng, thực hiện chức năng tiếp nhận, xử lý và phân phối dữ liệu trong toàn bộ hệ thống. Gateway nhận dữ liệu từ khói vi điều khiển thông qua giao thức MQTT, cung cấp dữ liệu cho Khối hiển thị và điều khiển và là nơi lưu lại thông tin của người dùng khi đăng ký trên khói hiển thị và điều khiển vào SQLite. Hơn nữa, gateway là nơi triển khai các cơ chế kiểm soát truy cập và bảo mật, đảm bảo các lệnh điều khiển được kiểm tra và xử lý trước khi chuyển xuống tầng thiết bị.
- Khối lưu trữ: có nhiệm vụ ghi nhận và quản lý dữ liệu vận hành của hệ thống, bao gồm dữ liệu cảm biến, trạng thái thiết bị và các sự kiện điều khiển. Dữ liệu được lưu trữ nhằm phục vụ việc giám sát lịch sử, phân tích và đánh giá hoạt động của hệ thống trong quá trình vận hành. Khối lưu trữ cũng cung cấp dữ liệu cho khói hiển thị và điều khiển khi người dùng truy cập hệ thống.
- Khối hiển thị và điều khiển: cung cấp giao diện tương tác giữa người dùng và hệ thống. Thông qua giao diện web, người dùng có thể theo dõi dữ liệu cảm biến theo thời gian thực, quan sát trạng thái hoạt động của các thiết bị và thực hiện các thao tác điều khiển từ xa. Hệ thống hỗ trợ cơ chế đăng ký và đăng nhập người dùng, trong đó sau khi đăng nhập, giao diện sẽ hiển thị các chức năng điều khiển được phân biệt theo vai trò người dùng. Cơ chế phân quyền này đảm bảo mỗi người dùng chỉ được phép truy cập và thao tác trong phạm vi chức năng phù hợp với quyền hạn được cấp.
- Khối nguồn gateway: cung cấp nguồn điện ổn định cho hệ thống gateway, đảm bảo Raspberry Pi và các thành phần liên quan hoạt động liên tục và tin cậy.

- Khối nguồn thiết bị: cung cấp năng lượng cho các thành phần tại tầng thiết bị, bao gồm khói vi điều khiển, khói cảm biến và khói điều khiển cơ cấu chấp hành. Nguồn điện được áp dụng về các mức điện áp phù hợp nhằm đảm bảo các khói hoạt động ổn định, an toàn và không bị ảnh hưởng bởi nhiễu hoặc dao động điện áp trong quá trình vận hành.

## 3.6 THIẾT KẾ PHẦN CỨNG

### 3.6.1 Thiết kế khói vi điều khiển

Khối vi điều khiển là thành phần trung tâm của tầng thiết bị, việc thiết kế khói vi điều khiển cho mô hình đòi hỏi phải đáp ứng đồng thời các yêu cầu về khả năng xử lý thời gian thực, độ ổn định khi vận hành liên tục, khả năng giao tiếp với nhiều loại thiết bị ngoại vi và tính linh hoạt để mở rộng trong tương lai. Cụ thể:

- Thứ nhất, khói vi điều khiển phải đảm nhận tốt việc thu thập và xử lý dữ liệu từ nhiều cảm biến môi trường khác nhau như pH, độ đục, nhiệt độ nước và mực nước, đồng thời thực thi các lệnh điều khiển mà không gây trễ hoặc xung đột trong quá trình vận hành.
- Thứ hai, trong khói vi điều khiển cần đảm bảo khả năng giao tiếp ổn định và chính xác giữa ESP WROM32 Module và STM32F103C8T6.
- Thứ ba, khói vi điều khiển phải hỗ trợ khả năng truyền thông mạng thông qua ESP32 WROM32 Module, cho phép trao đổi dữ liệu liên tục với tầng gateway bằng giao thức MQTT, đồng thời tiếp nhận và xử lý các lệnh điều khiển từ hệ thống giám sát mà vẫn đảm bảo tính an toàn và ổn định.

Khối vi điều khiển sử dụng STM32F103C8T6 (một vi điều khiển rất phổ biến trên thị trường dựa trên kiến trúc ARM Cortex-M3) và ESP32 WROM32 Module liên kết chặt chẽ với nhau để đáp ứng tốt các yêu cầu xử lý từ hệ thống.

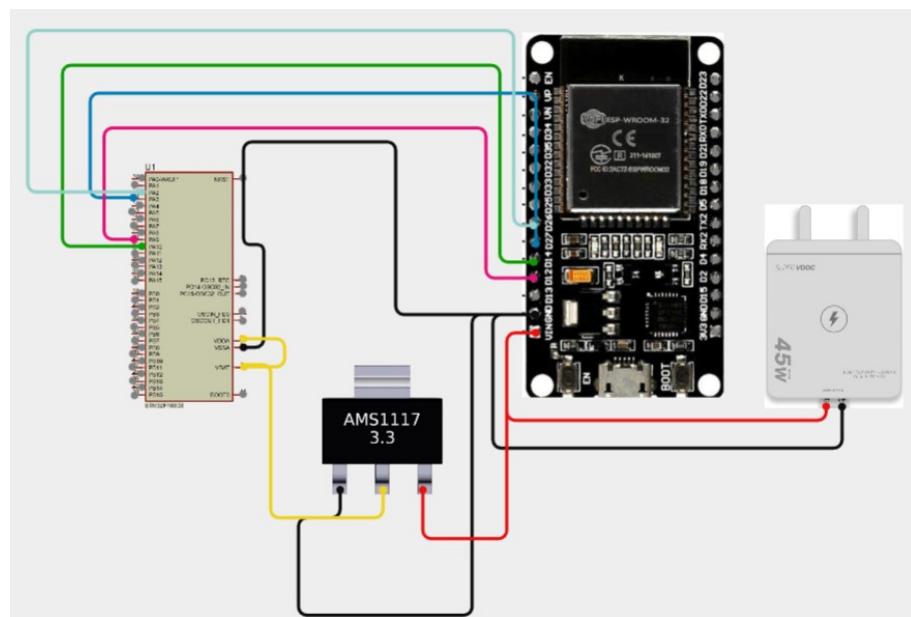
Cụ thể hơn khói này được mô tả chi tiết bởi các thành phần dưới đây:

- Vi điều khiển STM32F103C8T6.

- ESP32 WROOM32 Module.
- Giao tiếp:
  - + UART1: dùng cho đường dữ liệu cảm biến.
  - + UART2: dùng cho đường dữ liệu điều khiển.

Khối vi điều khiển được cấp nguồn từ adapter 5V 2A, trong đó vi điều khiển STM32 được cấp nguồn 3.3V từ AMS1117 và ESP32 nhận nguồn 5V vào chân Vin trên Module.

Dưới đây là sơ đồ kết nối của khói vi điều khiển:



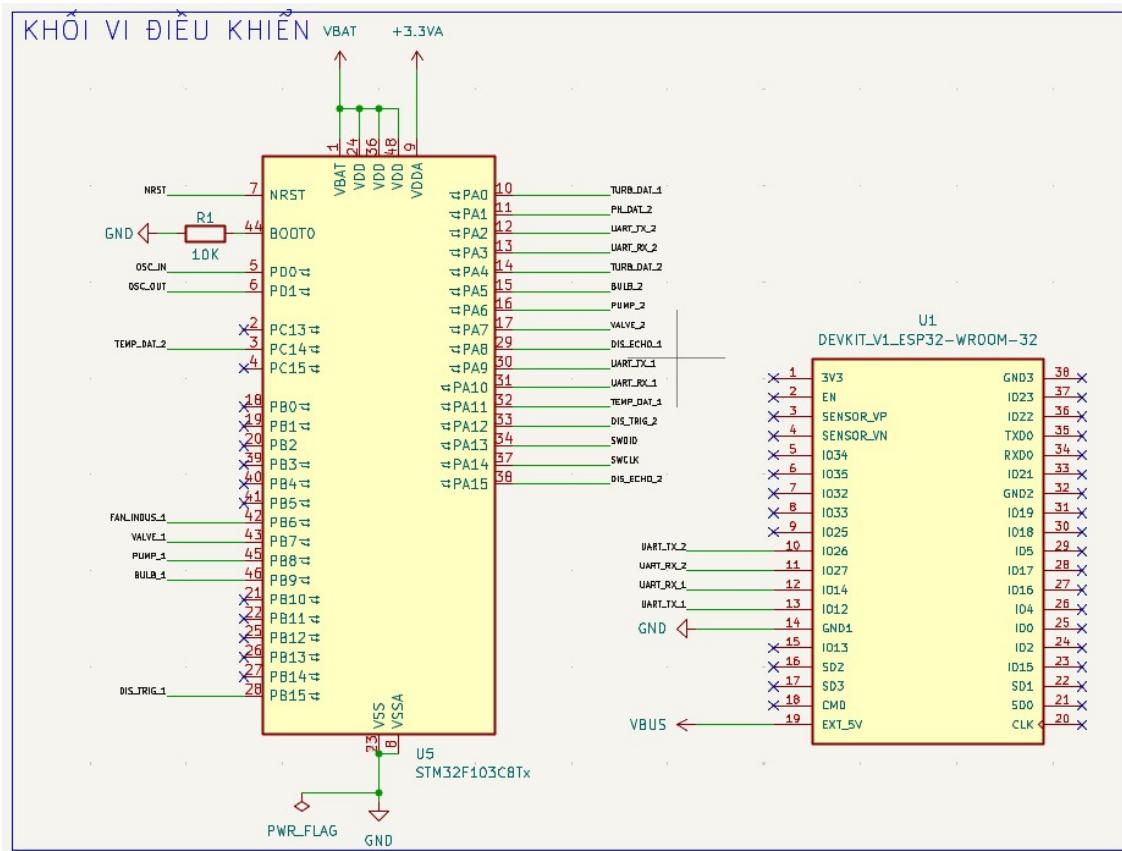
Hình 3.3: Sơ đồ kết nối của khói vi điều khiển

Tiếp theo là bảng trình bày sơ đồ nối chân của khói vi điều khiển:

Bảng 3.1: Sơ đồ kết nối giữa khối vi điều khiển và nguồn

<b>Nguồn</b>	<b>STM32</b>	<b>ESP32</b>	<b>AMS1117-3.3</b>
5V	5V	X	IN
GND	GND	GND	GND
x	PA9	G12	X
x	PA10	G14	X
x	PA2	G26	X
x	PA3	G27	X

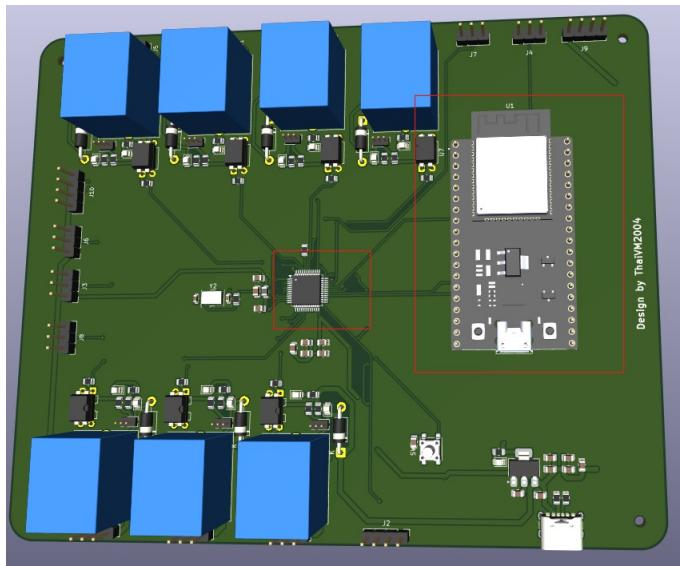
Sơ đồ nguyên lý của khối vi điều khiển:



Hình 3.4: Sơ đồ nguyên lý của khối vi điều khiển

Khi thiết kế mạch in cho khối vi điều khiển cần chú ý một số điểm sau:

- Vì điều khiển STM32 được bố trí tại trung tâm mạch in nhằm rút ngắn đường đi dây đến các cảm biến và module relay.
  - Module ESP32 WROM được đặt gần STM32 để giảm chiều dài đường giao tiếp UART, hạn chế nhiễu và suy hao tín hiệu.
  - Các tụ lọc nguồn được bố trí gần các chân cấp nguồn của STM32 nhằm đảm bảo nguồn điện ổn định trong quá trình hoạt động.



Hình 3.5: Vị trí của khối vi điều khiển trên mạch

Hình trên minh họa cách đặt khối vi điều khiển trên board mạch in.

### 3.6.2 Thiết kế khối cảm biến

Khối cảm biến là thành phần trực tiếp thu thập các thông số đặc trưng của môi trường thải tại khu vực giám sát. Khi tiến hành thiết kế khối cảm biến cần đảm bảo độ chính xác của dữ liệu đo, khả năng hoạt động ổn định, cũng như khả năng tương thích tốt với khối vi điều khiển. Cụ thể hơn, khối cảm biến cần đáp ứng các yêu cầu sau:

- Thứ nhất, khối cảm biến phải đo được các thông số đặc trưng của nước thải như pH, độ đục, nhiệt độ và mực nước.
- Thứ hai, tín hiệu ngõ ra của các cảm biến phải phù hợp với khả năng thu nhận và xử lý của vi điều khiển STM32, đảm bảo dữ liệu đo được ổn định và ít nhiễu.
- Thứ ba, khối cảm biến cần có khả năng hoạt động liên tục trong thời gian dài và cho phép hiệu chuẩn, thay thế dễ dàng trong quá trình vận hành và bảo trì.

Khối cảm biến trong mô hình sử dụng các loại cảm biến phổ biến, phù hợp với mục tiêu nghiên cứu và mô phỏng. Các cảm biến này được kết nối trực tiếp với vi

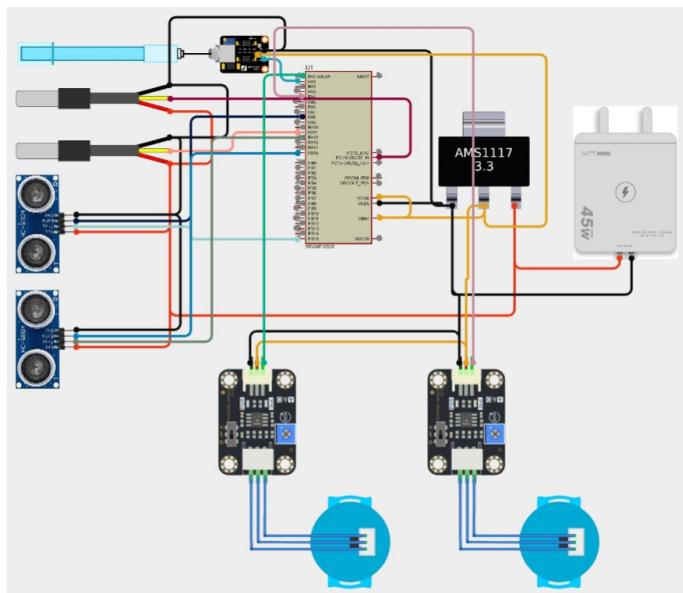
điều khiển STM32 để thu thập và xử lý dữ liệu theo thời gian thực.

Phần cứng của khối cảm biến bao gồm các thành phần sau:

- Cảm biến pH (ngõ ra analog).
- Cảm biến độ đục (ngõ ra analog).
- Cảm biến nhiệt độ DS18B20 (giao tiếp 1-Wire).
- Cảm biến siêu âm HC-SR04 để đo mực nước.

Dữ liệu từ các cảm biến được vi điều khiển STM32 đọc theo chu kỳ và thứ tự ưu tiên của các task được cấu hình theo FreeRTOS, sau đó được xử lý và truyền lên tầng gateway thông qua module ESP32 để phục vụ giám sát và điều khiển từ xa.

Dưới đây là sơ đồ kết nối của khối cảm biến với khối vi điều khiển:



Hình 3.6: Sơ đồ kết nối của khối cảm biến

Tiếp theo là các bảng trình bày kết nối trong khối cảm biến:

Bảng 3.2: Sơ đồ kết nối nguồn cảm biến

NGUỒN 5V-2A	AMS1117-3.3	PH	DS18B20	HCSR04	TURBIDITY
5V	IN	X	VCC	VCC	X
GND	GND	GND	GND	GND	GND
X	OUT	VCC	X	X	VCC

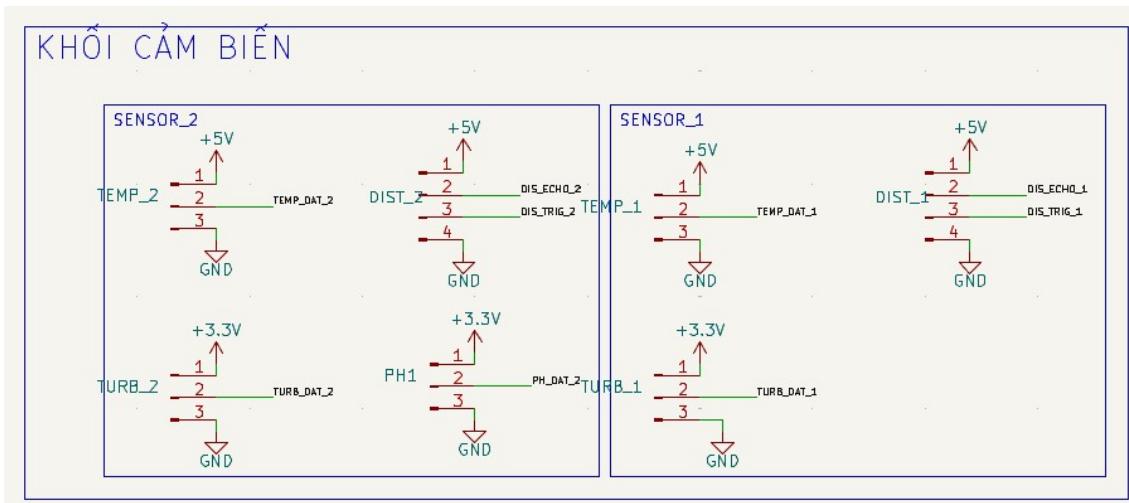
Bảng 3.3: Sơ đồ kết nối giữa khói cảm biến và vi điều khiển

<b>STM32</b>	<b>PH</b>	<b>DS18B20 1</b>	<b>DS18B20 2</b>	<b>TURBIDITY 1</b>	<b>TURBIDITY 2</b>
PA0	A	X	X	X	X
PA1	X	X	X	A	X
PA4	X	X	X	X	A
PA11	X	X	D	X	X
PC14	X	D	X	X	X

Bảng 3.4: Sơ đồ kết nối giữa khói cảm biến và vi điều khiển tiếp theo

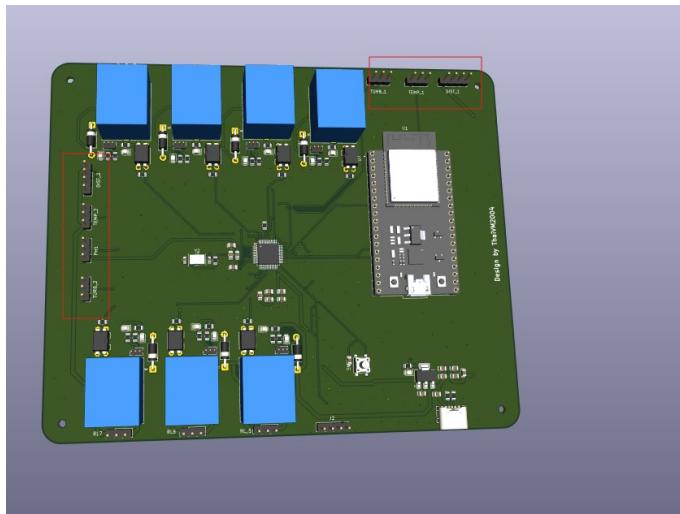
<b>STM32</b>	<b>TURBIDITY 1</b>	<b>TURBIDITY 2</b>
PA8	ECHO	X
PA12	X	ECHO
PA15	X	TRIG
PB15	TRIG	X

Sơ đồ nguyên lý của khói cảm biến:



Hình 3.7: Sơ đồ nguyên lý của khói cảm biến

Khi bố trí thiết kế lên mạch in, khói cảm biến là các hàng rào, các hàng rào này sẽ được bố trí sao cho thuận tiện cho việc kết nối với các cảm biến, và tránh đặt gần dây nguồn vì khả năng nhiễu điện áp xảy ra rất cao với các tín hiệu thuộc dãy tần số cao.



Hình 3.8: Vị trí của khối cảm biến trên mạch

### 3.6.3 Thiết kế khói điều khiển cơ cấu chấp hành

Khối điều khiển cơ cấu chấp hành có nhiệm vụ thực thi các lệnh điều khiển do khói vi điều khiển phát ra. Việc thiết kế khói điều khiển cơ cấu chấp hành cho mô hình cần đảm bảo khả năng đóng cắt ổn định, an toàn cho vi điều khiển và thuận tiện cho việc mô phỏng cũng như mở rộng trong quá trình nghiên cứu. Cụ thể, khói điều khiển cơ cấu chấp hành cần đáp ứng các yêu cầu sau:

- Thứ nhất, khói này phải cho phép vi điều khiển điều khiển bật/tắt các thiết bị relay mô phỏng một cách chính xác và tin cậy cao.
- Thứ hai, khói điều khiển cần đảm bảo cách ly và bảo vệ vi điều khiển khỏi các nhiễu và xung điện áp phát sinh.

Cụ thể, phần cứng của khói điều khiển cơ cấu chấp hành bao gồm:

- Các relay đơn điều khiển bằng tín hiệu số.
- Các chân GPIO của vi điều khiển STM32 dùng để điều khiển relay.
- Nguồn cấp 5V cho module relay.
- Transistor NPN 2N2222 dùng làm tầng khuếch đại dòng cho relay.

- Opto cách ly PC817 nhằm cách ly điện giữa khối điều khiển và khối công suất.
- Điện trở hạn dòng  $1\text{ k}\Omega$  cho LED opto và các LED chỉ thị.
- LED báo nguồn hoạt động của khối điều khiển.
- LED báo trạng thái kích relay.

Việc lựa chọn các linh kiện trong khối điều khiển cơ cấu chấp hành được thực hiện dựa trên yêu cầu về mức điện áp, dòng điện điều khiển, khả năng cách ly và độ tin cậy của hệ thống. Các bước tính toán và lựa chọn linh kiện được tiến hành như sau.

Trước hết, tín hiệu điều khiển xuất ra từ chân GPIO của vi điều khiển STM32 có mức điện áp 3.3 V và khả năng cấp dòng giới hạn, không đủ để kích trực tiếp cuộn dây relay. Do đó, transistor NPN 2N2222 được sử dụng làm phần tử khuếch đại dòng. Relay sử dụng trong mô hình có điện áp cuộn dây 5 V và dòng kích khoảng vài chục mA, trong khi 2N2222 có khả năng chịu dòng collector lên đến vài trăm mA, đáp ứng tốt yêu cầu điều khiển.

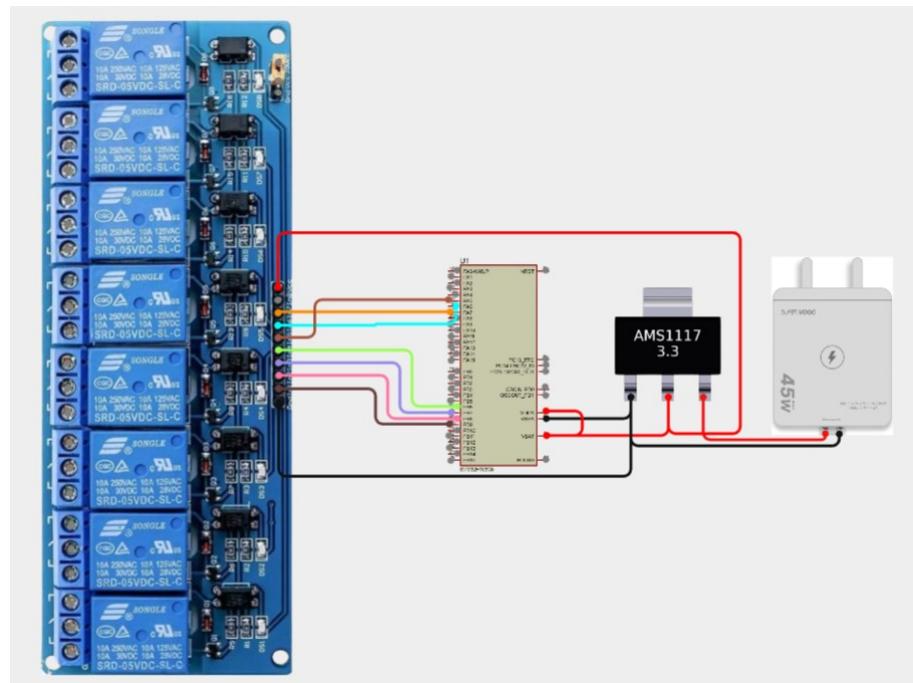
Để đảm bảo an toàn và giảm nhiễu từ phía relay quay ngược về vi điều khiển, opto cách ly PC817 được sử dụng giữa GPIO và tầng khuếch đại. PC817 cho phép cách ly quang hoàn toàn giữa mạch điều khiển mức thấp và mạch công suất, qua đó nâng cao độ ổn định và độ tin cậy của hệ thống.

Điện trở hạn dòng  $1\text{ k}\Omega$  được lựa chọn cho LED bên trong opto PC817 và các LED chỉ thị. Với mức điện áp điều khiển 3.3 V từ GPIO, điện áp rơi trên LED khoảng 1.2 V đến 2.0 V, dòng qua LED vào khoảng 1–2 mA, đảm bảo LED hoạt động ổn định, đủ sáng và không vượt quá khả năng cấp dòng của vi điều khiển.

Các LED báo nguồn và LED báo trạng thái relay được bổ sung nhằm hỗ trợ quan sát trực quan trạng thái hoạt động của khối điều khiển trong quá trình vận hành và thí nghiệm. Việc này giúp người vận hành dễ dàng kiểm tra tình trạng cấp nguồn cũng như xác nhận tín hiệu điều khiển đã được thực thi.

Qua quá trình tính toán và lựa chọn linh kiện như trên, khôi điều khiển cơ cấu chấp hành đảm bảo đáp ứng yêu cầu về khả năng điều khiển, độ an toàn điện và tính ổn định khi tích hợp vào hệ thống IoT giám sát và điều khiển xử lý nước thải.

Dưới đây là sơ đồ kết nối của khôi điều khiển cơ cấu chấp hành trong giai đoạn thử nghiệm mô hình.

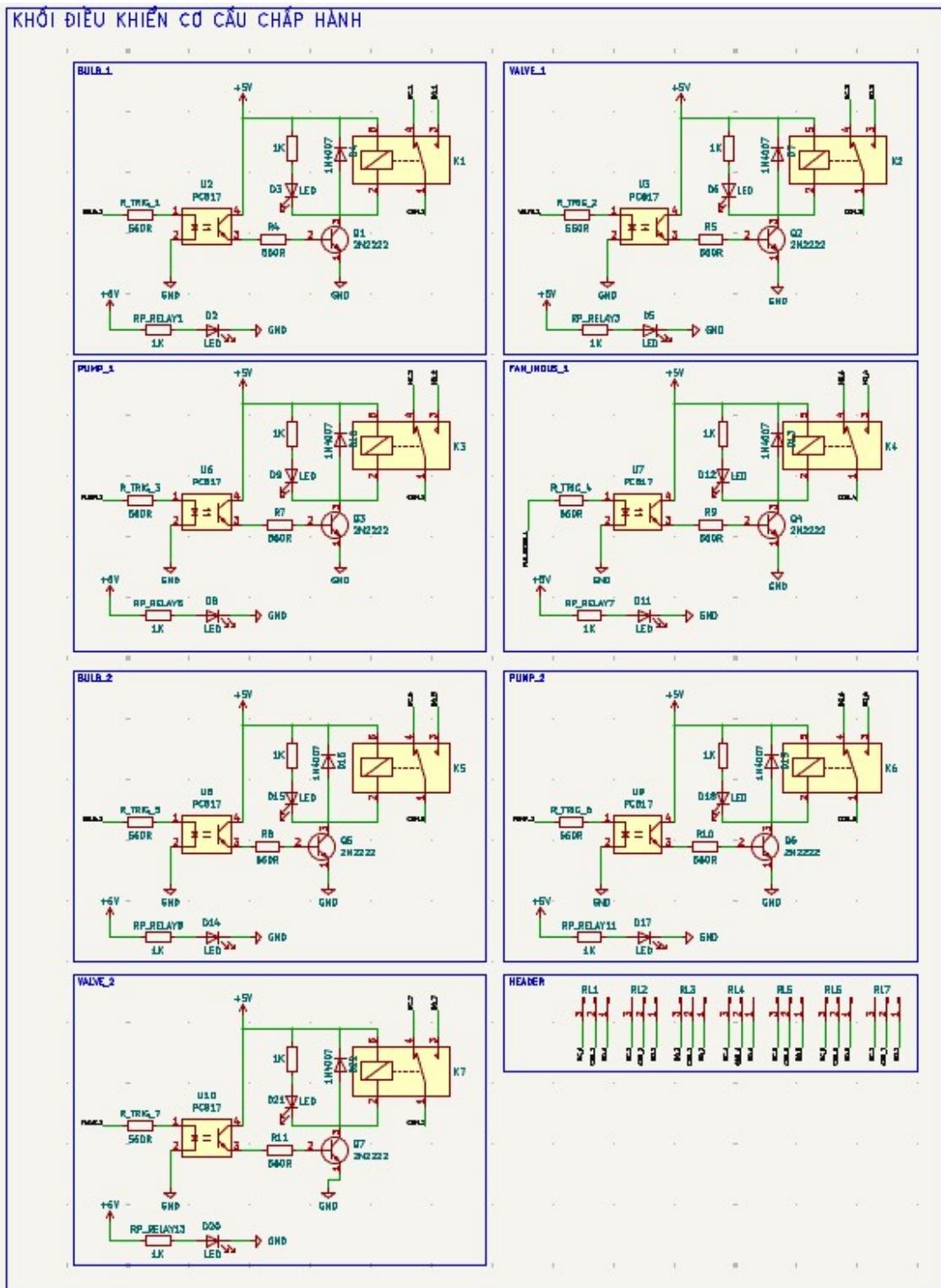


Hình 3.9: Sơ đồ kết nối của khôi điều khiển cơ cấu chấp hành

Bảng 3.5: Sơ đồ kết nối giữa khối điều khiển cơ cầu chìp hành và vi điều khiển

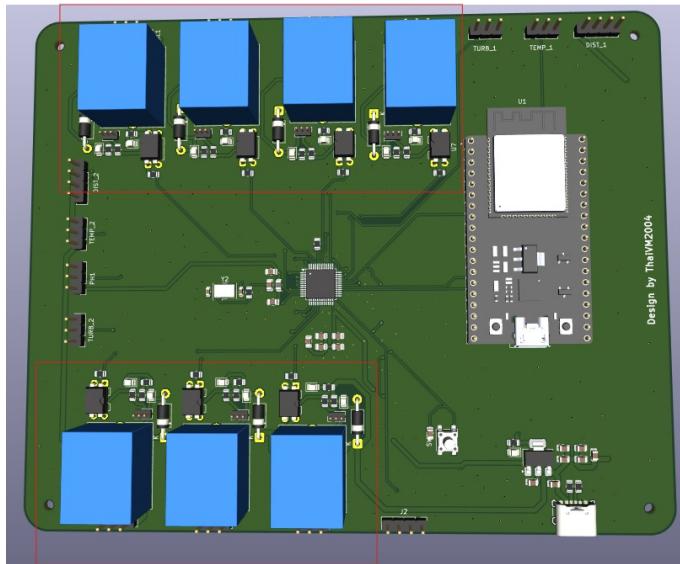
<b>5V</b>	<b>AMS1117</b>	<b>STM32</b>	<b>Relay</b>	<b>Chức năng mô phỏng</b>
5V	IN	X	X	X
GND	GND	GND	X	X
X	OUT	VCC	X	X
X	X	PA5	IN5	Đèn 2
X	X	PA6	IN6	Bơm 2
X	X	PA7	IN7	Van 2
X	X	PB6	IN4	Động cơ khuấy 1
X	X	PB7	IN3	Van 1
X	X	PB8	IN2	Bơm 1
X	X	PB9	IN1	Đèn 1

Tiếp theo là sơ đồ nguyên lý của khối điều khiển cơ cầu chìp hành:



Hình 3.10: Sơ đồ nguyên lý khói điều khiển cơ cấu chấp hành

Trong thiết kế mạch in, khói điều khiển cơ cấu chấp hành được bố trí tách biệt tương đối với khói cảm biến và khói vi điều khiển nhằm hạn chế ảnh hưởng của nhiễu điện từ từ các đường cấp nguồn và đóng cắt tải.



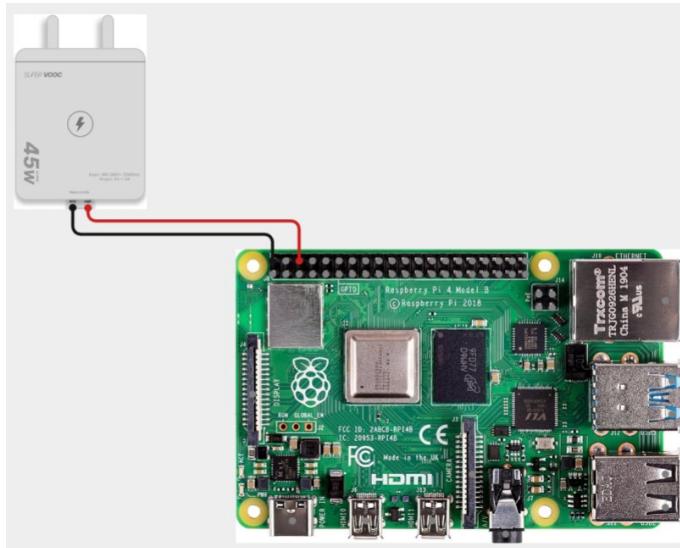
Hình 3.11: Vị trí của khối điều khiển cơ cầu chìp hành trên mạch

### 3.6.4 Thiết kế khối gateway

Khối gateway là thành phần quan trọng của hệ thống, đóng vai trò cầu nối trung gian giữa tầng thiết bị và tầng ứng dụng. Khi tiến hành thiết kế khói này cần đáp ứng các yêu cầu về khả năng xử lý liên tục, độ ổn định cao khi vận hành trong thời gian dài, cũng như đảm bảo an toàn và bảo mật khi hệ thống được kết nối mạng. Cụ thể, khói gateway cần đáp ứng các yêu cầu sau:

- Thứ nhất, khói gateway phải được thiết kế với nguồn cấp ổn định, đáp ứng đúng điện áp và dòng yêu cầu của Raspberry Pi 4, đảm bảo hệ thống có thể hoạt động liên tục trong thời gian dài mà không xảy ra sụt áp hoặc treo hệ thống.
- Thứ hai, phần cứng gateway cần đảm bảo khả năng hoạt động bền bỉ trong điều kiện vận hành 24/7, có khả năng tản nhiệt tốt và hạn chế quá nhiệt trong quá trình xử lý dữ liệu liên tục.
- Thứ ba, phần cứng gateway cần đảm bảo độ tin cậy cao, hạn chế lỗi do rung động, mất nguồn đột ngột hoặc nhiễu điện, phù hợp với môi trường vận hành công nghiệp mô phỏng.

Dưới đây là sơ đồ kết nối với nguồn của khói gateway trong hệ thống:



Hình 3.12: Sơ đồ kết nối của khói gateway với nguồn

Tiếp theo là sơ đồ nối chân của khói gateway với nguồn adapter:

Bảng 3.6: Sơ đồ kết nối của khói gateway

5V	RASPBERRY PI 4
5V	5V
GND	GND

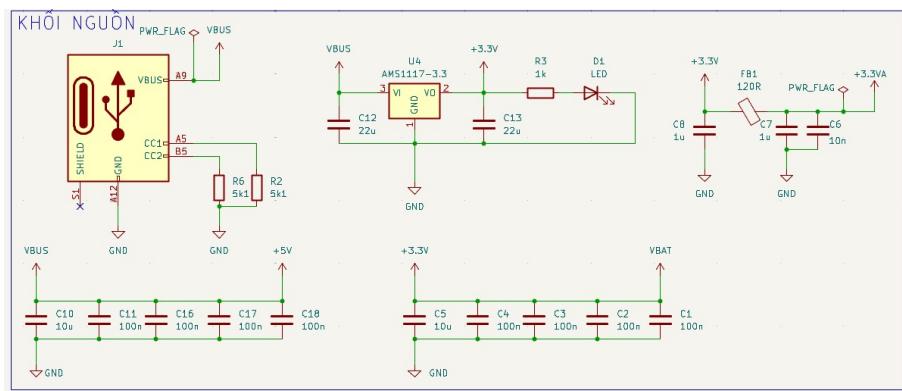
Trong quá trình bố trí hệ thống, khói gateway được đặt tách biệt với các khói thiết bị tại hiện trường nhằm phục vụ cho chức năng giám sát và điều khiển từ xa. Cách bố trí này giúp khói Gateway hoạt động ổn định, hạn chế ảnh hưởng của nhiều và tạo thuận lợi cho công tác bảo trì cũng như mở rộng hệ thống trong quá trình triển khai.

### 3.6.5 Thiết kế khói nguồn thiết bị

Khối nguồn thiết bị là thành phần quan trọng nhất của tầng thiết bị. Trong dự án này, khói nguồn thiết bị được thiết kế phải đảm bảo các nhiệm vụ sau đây:

- Cung cấp đầy đủ và ổn định các mức điện áp cần thiết cho khói vi điều khiển, khói cảm biến, khói điều khiển cơ cấu chấp hành.
  - Đảm bảo công suất đầu ra phù hợp với mức tiêu thụ cao nhất của tầng thiết bị, đặc biệt là khi kết nối Wifi trên ESP32.
  - Tổ chức lọc nhiễu nguồn, tránh hiện tượng sụt áp, nhiễu chéo giao thức UART, các giao cảm biến với vi điều khiển.
  - Đảm bảo an toàn, dễ thay thế, dễ kiểm soát trong quá trình vận hành và bảo trì

Dưới đây là sơ đồ nguyên lý của Khối nguồn thiết bị:



Hình 3.13: Sơ đồ nguyên lý khôi nguồn thiết bị

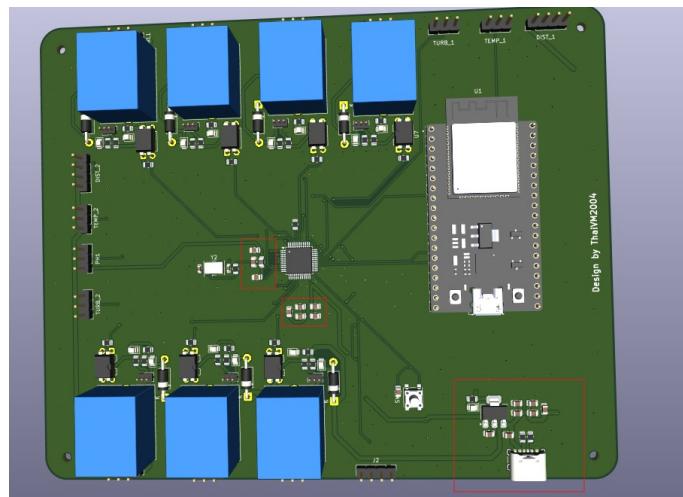
Tầng thiết bị sẽ sử dụng nguồn 5V từ cổng USB Type C (6 chân), thông qua đường VBUS, từ đây nguồn được lọc qua các tụ và phân phối đến nhiều nơi khác nhau.

Vai trò của các tụ lọc nhiễu:

- Các tụ C10, C5, C6, có giá trị 10n nhiệm vụ là lọc nhiễu tầng số cao từ vài MHz.
  - Các tụ có giá trị 100n có thể lọc nhiễu ở dãy tầng từ vài chục MHz.
  - Còn lại các tụ có giá trị từ vài  $\mu$  đến vài chục  $\mu$  có thể lọc nhiễu nguồn DC tầng số thấp Hz.

Sau khi lọc xong thì nguồn 5V sẽ qua IC nguồn AMS1117-3.3V để biến thành điện áp 3.3V nuôi nguồn cho vi điều khiển STM32 và các cảm biến hoạt động ở điện áp 3.3V như cảm biến pH, cảm biến độ đục. Ngoài ra VBUS này sẽ cấp trực tiếp cho chân 5V trên ESP32 Module vì trên đây đã tích hợp sẵn IC nguồn.

Khi thiết kế mạch in cần lưu ý, các tụ lọc nguồn 3.3V nên đặt gần STM32F103C8T6 để hạn chế hiện tượng sụt áp khi dây đi quá dài. USB cấp nguồn thì nên đặt gần outline của board để dễ dàng kết nối với cáp cấp nguồn. Cụ thể hình dưới đây trình bày những yêu cầu trên:



Hình 3.14: Vị trí đặt của khối nguồn thiết bị trên mạch in

Để chọn được nguồn có công suất phù hợp, đồng thời cấp ổn định cho toàn bộ hệ thống, ta phải đi tính toán công suất từ các tải tiêu hao.

Bảng 3.7: Công suất tiêu thụ cực đại của tầng thiết bị

Thiết bị	Số lượng	Điện áp (V)	Dòng cực đại (mA)	Công suất (mW)
STM32F103C8T6	1	3.3	50	250
ESP32 Module	1	5	300	1500
DS18B20	2	5	10	100
Cảm biến pH	1	3.3	10	33
Cảm biến độ đục	2	3.3	80	264
Cảm biến khoảng cách HC-SR04	2	5	40	200
Relay 5V	7	5	490	2450
<b>Tổng</b>	x	x	<b>980</b>	<b>4797</b>

Như vậy, công suất tiêu thụ trên Tầng thiết bị khoảng  $4797\text{mW} = 4.797\text{W}$

Nguồn chính sử dụng là 5V-2A, nguồn có công suất 10W hoàn toàn đáp ứng cho tầng thiết bị kể cả khi tắt cả các thiết bị hoạt động đồng thời.

Khối nguồn là yếu tố nền tảng giúp toàn hệ thống hoạt động ổn định, bền vững. Việc thiết kế đúng kỹ thuật từ cách chọn nguồn, bố trí tụ lọc, đến đi dây PCB sẽ giúp giảm lỗi không mong muốn, tăng tuổi thọ phần cứng và tối ưu trải nghiệm sử dụng hệ thống trong thời gian dài.

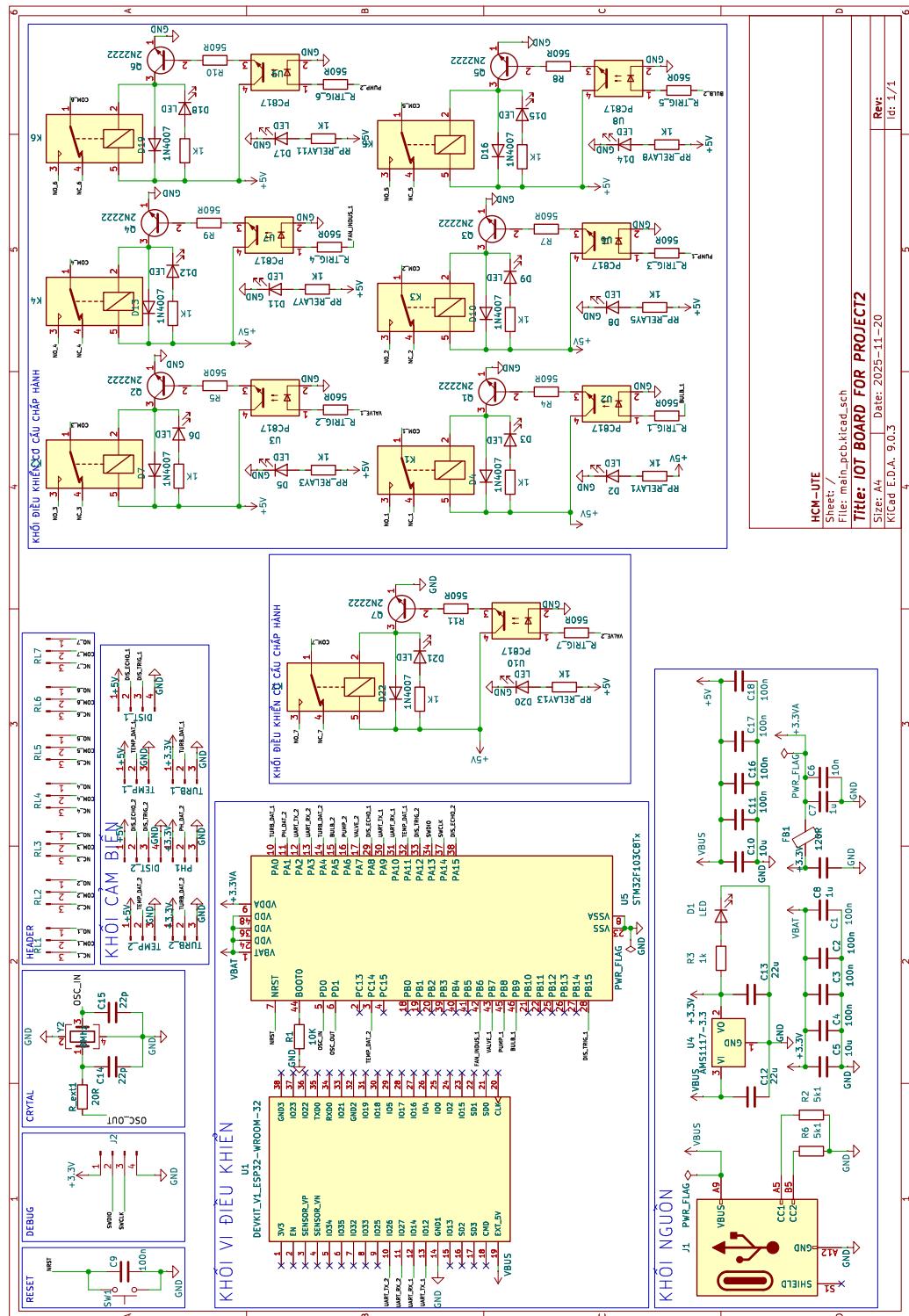
### 3.6.6 Thiết kế khôi nguồn gateway

Khối nguồn của gateway được thiết kế nhằm cung cấp điện áp ổn định và đủ công suất cho Raspberry Pi 4 trong suốt quá trình vận hành liên tục của hệ thống. Do gateway đóng vai trò trung tâm trong việc xử lý dữ liệu, giao tiếp mạng và điều phối hoạt động của các khôi khác, yêu cầu về nguồn cấp phải đảm bảo độ tin cậy cao, hạn chế sụt áp và đáp ứng tốt các điều kiện tải thay đổi.

Raspberry Pi 4 sử dụng nguồn một chiều 5V, cấp qua cổng USB-C, với dòng tiêu thụ có thể lên đến khoảng 3A khi hoạt động ở chế độ tải cao, bao gồm xử lý dữ liệu, truyền thông mạng và kết nối các thiết bị ngoại vi. Vì vậy, bộ nguồn được lựa chọn có điện áp đầu ra danh định 5V và khả năng cung cấp dòng tối thiểu 3A, nhằm đảm bảo hệ thống hoạt động ổn định, tránh hiện tượng quá dòng, sụt áp hoặc tự khởi động lại trong quá trình vận hành.

Việc lựa chọn nguồn đáp ứng đúng thông số kỹ thuật của Raspberry Pi 4 giúp nâng cao độ ổn định và độ tin cậy của gateway, đồng thời đảm bảo hệ thống có thể hoạt động liên tục trong thời gian dài, phù hợp với yêu cầu của mô hình giám sát và điều khiển trong môi trường công nghiệp.

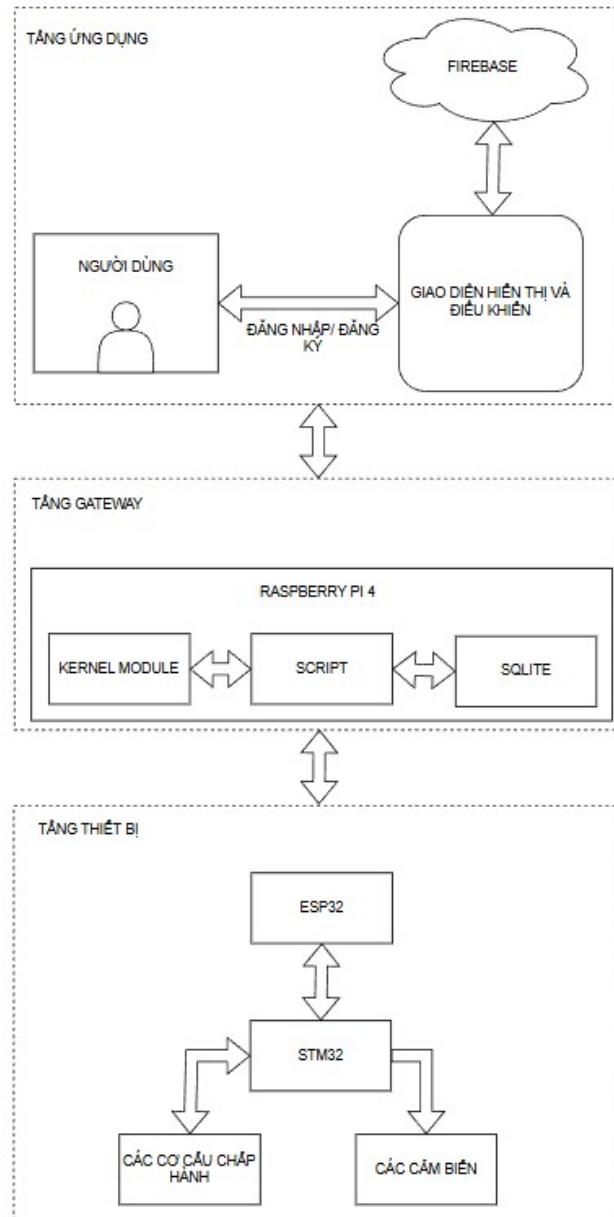
Từ các thiết kế chi tiết của từng khôi chức năng, bước tiếp theo là xây dựng sơ đồ nguyên lý tổng thể (schematic) của hệ thống. Sơ đồ này thể hiện mối liên kết giữa các khôi phần cứng, luồng tín hiệu điều khiển và nguồn cấp, làm cơ sở cho việc thi công mô hình và triển khai các chức năng điều khiển – giám sát trong các phần tiếp theo của đề tài.



Hình 3.15: Sơ nguyên lý toàn bộ hệ thống

## 3.7 THIẾT KẾ PHẦN MỀM

### 3.7.1 Mô tả luồng hoạt động của hệ thống



Hình 3.16: Sơ đồ mô tả chi tiết hệ thống

Từ sơ đồ mô tả chi tiết trên ta có thể dựa trên cơ sở đó phân tích luồng hoạt động tổng thể của hệ thống phần mềm, thể hiện mối liên kết giữa các thành phần chính bao gồm: người dùng, giao diện hiển thị – điều khiển, các dịch vụ Firebase, tầng gateway và tầng thiết bị.

Trước hết, người dùng thực hiện thao tác đăng ký hoặc đăng nhập thông qua giao diện hiển thị và điều khiển. Thông tin đăng ký và xác thực được chuyển tới dịch vụ Firebase Authentication để kiểm tra tính hợp lệ. Sau khi xác thực thành công, người dùng được cấp quyền truy cập tương ứng với vai trò (Admin, User1 hoặc User2) theo như thông tin đã đăng ký, và giao diện sẽ hiển thị các chức năng phù hợp.

Trong quá trình vận hành, dữ liệu cảm biến từ tầng thiết bị được thu thập bởi vi điều khiển STM32, sau đó truyền lên ESP32. ESP32 đóng vai trò trung gian truyền thông, gửi dữ liệu cảm biến đã thu thập lên khối gateway thông qua kết nối mạng. Gateway tiếp nhận dữ liệu, thực hiện các bước kiểm tra cần thiết và lưu trữ dữ liệu vào Firebase Realtime Database nhằm phục vụ hiển thị thời gian thực trên giao diện người dùng.

Song song với đó, các lệnh điều khiển do người dùng thao tác trên giao diện được ghi nhận và lưu trữ tại Cloud Firestore. Khối gateway định kỳ kiểm tra các lệnh mới, tiến hành xác thực thông tin và trạng thái đăng ký của thiết bị thông qua cơ sở dữ liệu SQLite cục bộ. Sau khi lệnh được kiểm tra hợp lệ, gateway chuyển tiếp lệnh điều khiển xuống ESP32 và STM32 để thực thi trên các thiết bị chấp hành.

Với cơ chế trao đổi dữ liệu hai chiều và sự kết hợp giữa các nền tảng lưu trữ, hệ thống đảm bảo tính đồng bộ, an toàn và bảo mật.

### 3.7.2 Thiết kế cơ chế bảo mật đa tầng

Ý tưởng của cơ chế bảo mật đa tầng trong dự án này là phân tách rõ ràng các lớp bảo vệ theo từng tầng kiến trúc, bao gồm tầng ứng dụng, tầng gateway và tầng thiết bị. Mỗi tầng đảm nhiệm một vai trò bảo mật riêng, giúp giảm thiểu rủi ro khi một tầng bị tấn công hoặc gặp sự cố, đồng thời tăng khả năng kiểm soát và truy vết các thao tác trong toàn bộ hệ thống.

Dựa trên đặc thù của mô hình IoT giám sát và điều khiển xử lý nước thải công nghiệp, cơ chế bảo mật đa tầng cần đáp ứng các yêu cầu chính sau:

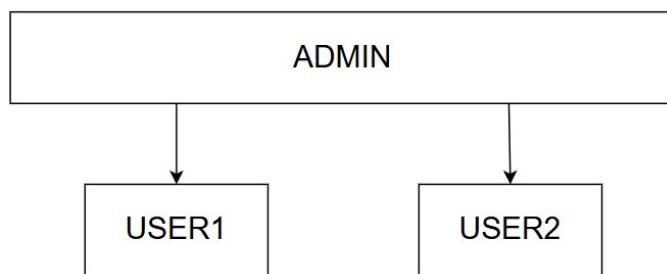
- Đảm bảo chỉ người dùng hợp lệ mới có thể truy cập hệ thống thông qua cơ

chế xác thực và phân quyền rõ ràng.

- Ngăn chặn các lệnh điều khiển trái phép hoặc không đúng quyền truy cập từ tầng ứng dụng xuống tầng thiết bị.
- Kiểm soát luồng dữ liệu và lệnh điều khiển tại gateway nhằm hạn chế nguy cơ tấn công từ mạng Internet.
- Đảm bảo các thiết bị nhúng chỉ thực thi các lệnh đã được xác thực và kiểm tra tính hợp lệ.
- Hỗ trợ ghi nhận và truy vết các thao tác điều khiển phục vụ công tác giám sát và phân tích sự cố.

### Cơ chế bảo mật tại tầng ứng dụng

Tại tầng ứng dụng, hệ thống sử dụng Firebase Authentication để thực hiện xác thực người dùng thông qua tài khoản đăng nhập. Mỗi người dùng được gán một vai trò cụ thể, từ đó giới hạn quyền truy cập và thao tác điều khiển trên giao diện.



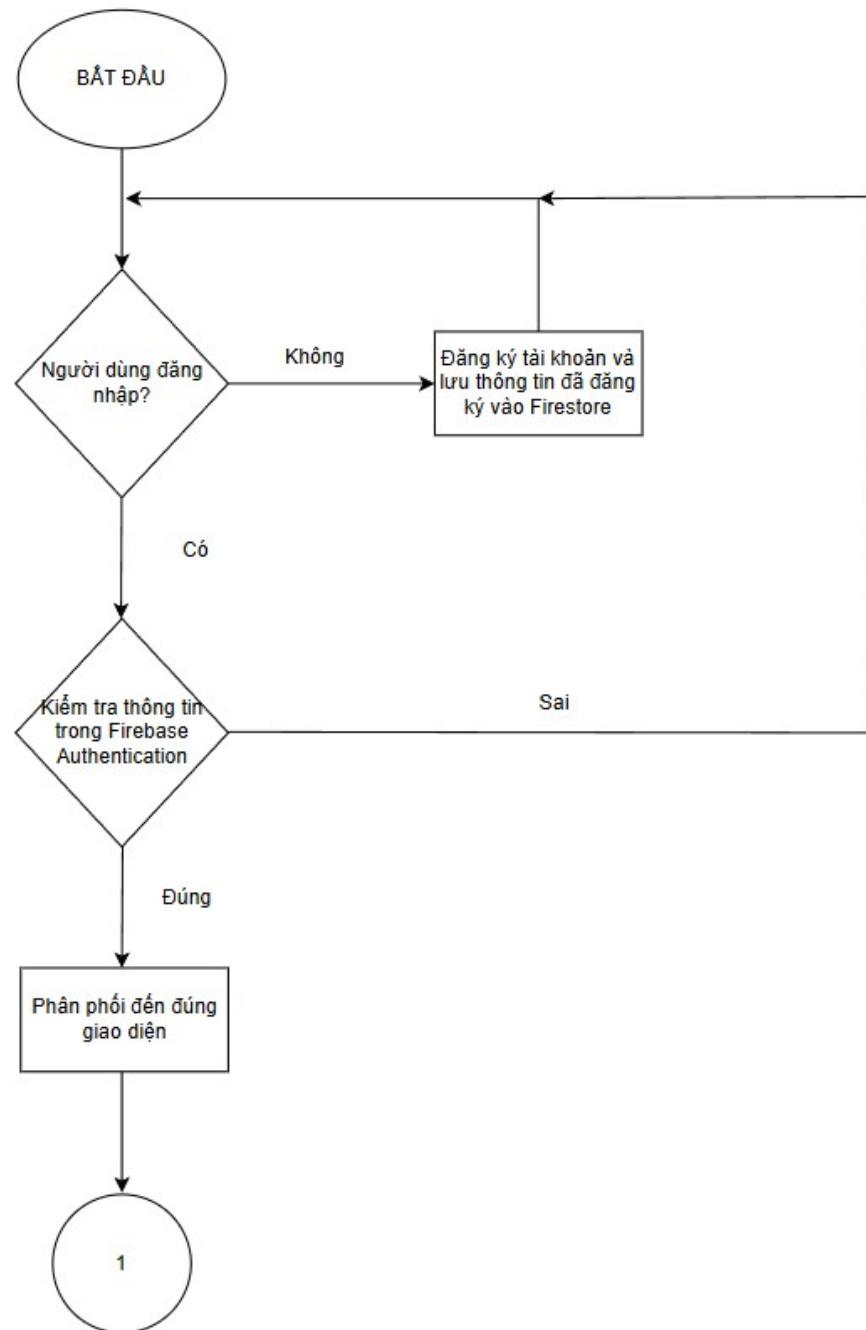
Hình 3.17: Cơ chế phân quyền người dùng

Trong đó:

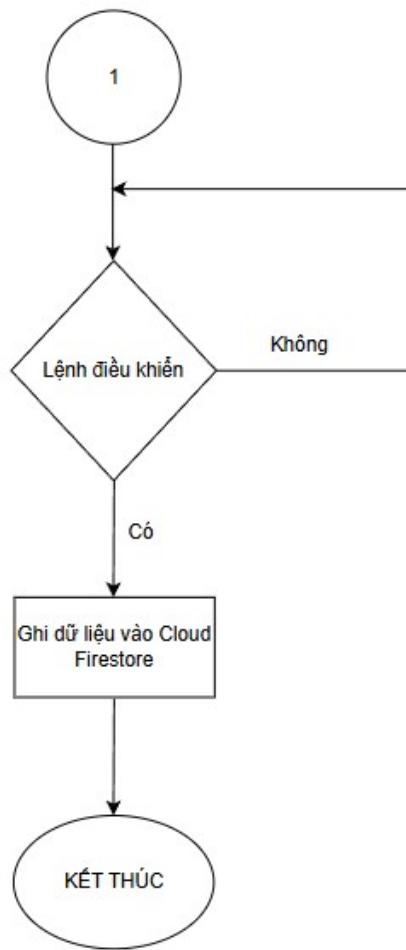
- Admin: có quyền điều khiển và kiểm soát toàn hệ thống.
- User1: có quyền điều khiển các thiết bị cảm biến tại bể thu gom (Collection Tank).
- User2: có quyền điều khiển các thiết bị cảm biến tại bể đầu ra (Discharge Tank).

Các thao tác điều khiển từ giao diện không được gửi trực tiếp xuống thiết bị, mà được ghi nhận và lưu trữ tại Cloud Firestore. Điều này giúp tách biệt rõ ràng giữa tầng giao diện và tầng điều khiển vật lý, đồng thời tạo điều kiện cho việc kiểm tra và truy vết lệnh.

Dưới đây là lưu đồ thuật toán minh họa cho quá trình bảo mật tại tầng ứng dụng:



Hình 3.18: Lưu đồ thuật toán quá trình bảo mật ở tầng ứng dụng



Hình 3.19: Lưu đồ thuật toán quá trình bảo mật ở tầng ứng dụng - tiếp theo

Mô tả cơ chế: Lưu đồ trên mô tả cơ chế bảo mật được triển khai tại tầng ứng dụng của hệ thống, tập trung vào quá trình xác thực người dùng và kiểm soát lệnh điều khiển. Ở giai đoạn đầu, hệ thống kiểm tra trạng thái đăng nhập của người dùng. Trường hợp người dùng chưa có tài khoản, hệ thống cho phép đăng ký và lưu trữ thông tin đăng ký vào Cloud Firestore. Khi người dùng thực hiện đăng nhập, thông tin xác thực sẽ được kiểm tra thông qua Firebase Authentication nhằm đảm bảo tính hợp lệ trước khi cho phép truy cập.

Sau khi xác thực thành công, người dùng được phân phối đến giao diện tương ứng với vai trò và quyền truy cập đã được cấp. Tại đây, các thao tác điều khiển của người dùng tiếp tục được giám sát. Mỗi lệnh điều khiển hợp lệ sẽ được ghi nhận và lưu trữ trên Cloud Firestore trước khi chuyển sang các tầng xử lý tiếp theo. Cơ

chế này giúp đảm bảo rằng chỉ các người dùng đã được xác thực và các lệnh hợp lệ mới được hệ thống chấp nhận, đồng thời hỗ trợ truy vết và kiểm soát hoạt động điều khiển trong toàn bộ quá trình vận hành.

Tại Cloud Firestore thông tin sẽ được ghi lại, trong đó hai trường quan trọng nhất là:

- command: có cấu trúc "db/username/rule/device/status"
- processed: nhận giá trị true hoặc false

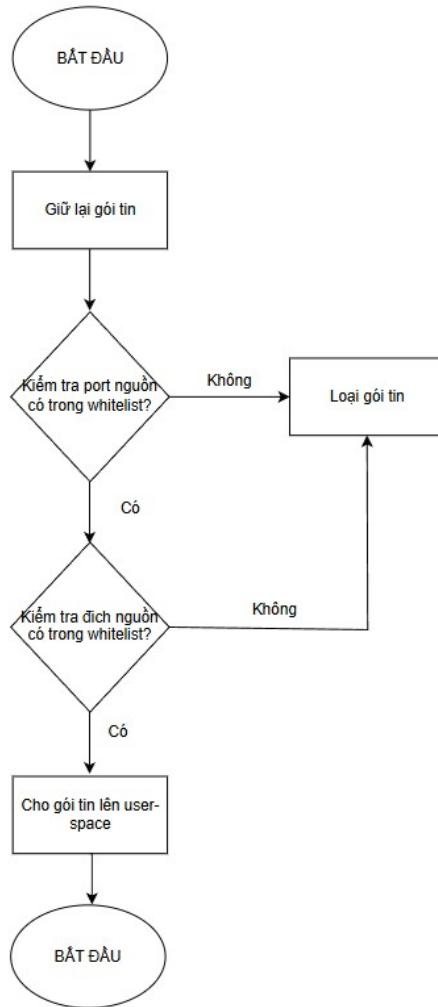
Hai thông tin trên sẽ được đề cập chi tiết ở phần thiết kế phần mềm khôi lưu trữ.

### **Cơ chế bảo mật tại tầng gateway**

Gateway đóng vai trò là lớp bảo vệ trung tâm của hệ thống. Tất cả dữ liệu cảm biến và lệnh điều khiển đều phải đi qua gateway trước khi được xử lý hoặc chuyển tiếp. Gateway thực hiện kiểm tra tính hợp lệ của lệnh điều khiển dựa trên thông tin người dùng, vai trò và trạng thái đăng ký thiết bị.

Trên gateway cơ chế bảo mật được chia thành 2 lớp là lớp kernel và lớp user cụ thể thuật toán được trình bày dưới đây:

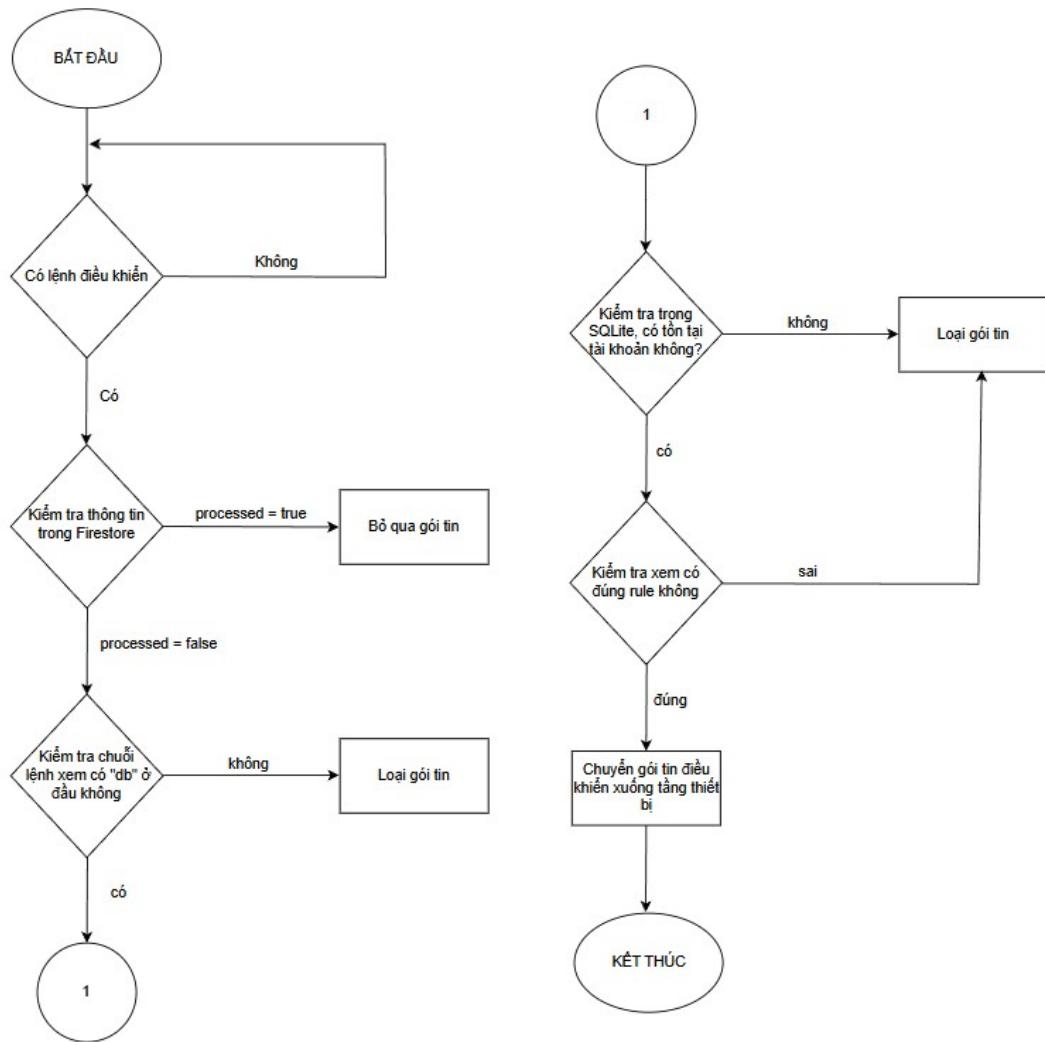
Thuật toán ở kernel-space:



Hình 3.20: Lưu đồ thuật toán quá trình bảo mật ở gateway tại kernel-space

Khi một gói tin mạng đi vào gateway, module bảo mật trong kernel tiến hành kiểm tra thông tin gói tin như địa chỉ nguồn và cổng nguồn. Dựa trên các luật lọc được cấu hình sẵn trong Netfilter, gói tin sẽ được phân loại là hợp lệ hoặc không hợp lệ. Đối với các gói tin hợp lệ, kernel cho phép chuyển tiếp lên tầng user-space để tiếp tục xử lý. Ngược lại, các gói tin không hợp lệ hoặc có dấu hiệu bất thường sẽ bị chặn ngay tại kernel-space.

Thuật toán ở user-space:



Hình 3.21: Lưu đồ thuật toán quá trình bảo mật ở gateway tại user-space

Hình minh họa quy trình xử lý bảo mật tại tầng user-space của gateway, tập trung vào việc xác thực và kiểm soát các yêu cầu đến từ tầng ứng dụng. Sau khi gói tin vượt qua lớp kiểm tra ở kernel-space, dữ liệu được chuyển lên user-space để tiến hành xác thực thông tin người dùng và thiết bị.

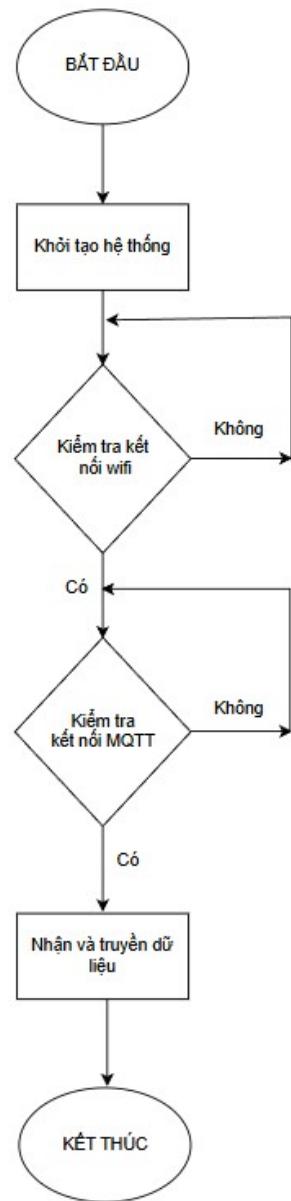
Tại đây, gateway kiểm tra trạng thái đăng ký, thông tin định danh và quyền truy cập tương ứng. Các yêu cầu không hợp lệ, chưa đăng ký hoặc không đủ quyền sẽ bị từ chối.

### Cơ chế bảo mật tại tầng thiết bị

Tại tầng thiết bị, vi điều khiển ESP32 chỉ thực thi các lệnh điều khiển được gửi từ gateway sau khi đã được xác thực. Thiết bị không cho phép nhận lệnh trực tiếp

từ Internet hoặc từ giao diện người dùng, qua đó giảm thiểu nguy cơ bị can thiệp trái phép.

Dữ liệu cảm biến được gửi ngược lên gateway theo một chiều, đảm bảo thiết bị chỉ đóng vai trò thu thập và thực thi, không lưu trữ thông tin nhạy cảm liên quan đến người dùng hay quyền truy cập.



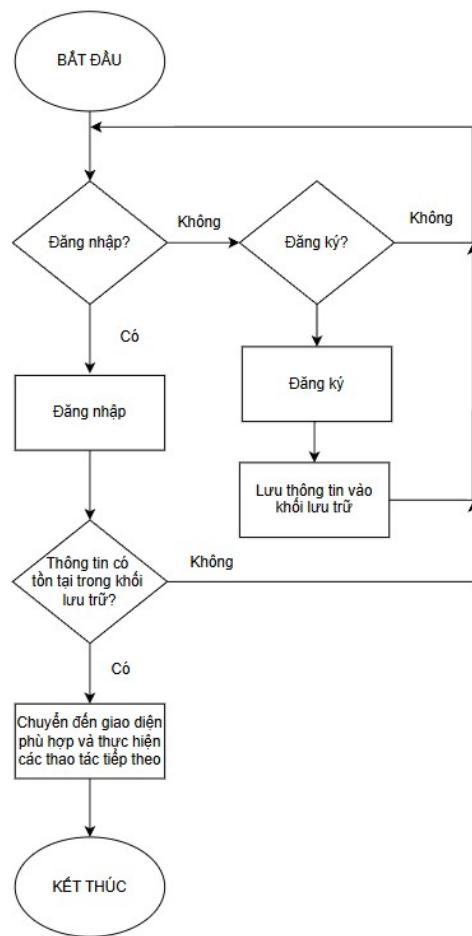
Hình 3.22: Lưu đồ thuật toán quá trình bảo mật ở tầng thiết bị

### 3.7.3 Thiết kế phần mềm tầng ứng dụng

Tầng ứng dụng đóng vai trò là lớp tương tác trực tiếp với người dùng, thực hiện giám sát trạng thái hệ thống, hiển thị dữ liệu cảm biến thời gian thực và gửi lệnh điều khiển thông qua Gateway. Tầng này đảm bảo tính minh bạch của dữ liệu và thực thi cơ chế phân quyền bảo mật.

Chương trình trong tầng ứng dụng được thiết kế chia thành ba quá trình chính bao gồm: quá trình xác thực dữ liệu người dùng, quá trình gửi dữ liệu đến tầng gateway, quá trình nhận dữ liệu từ tầng gateway.

Đầu tiên là lưu đồ trình bày quá trình xác thực dữ liệu người dùng:

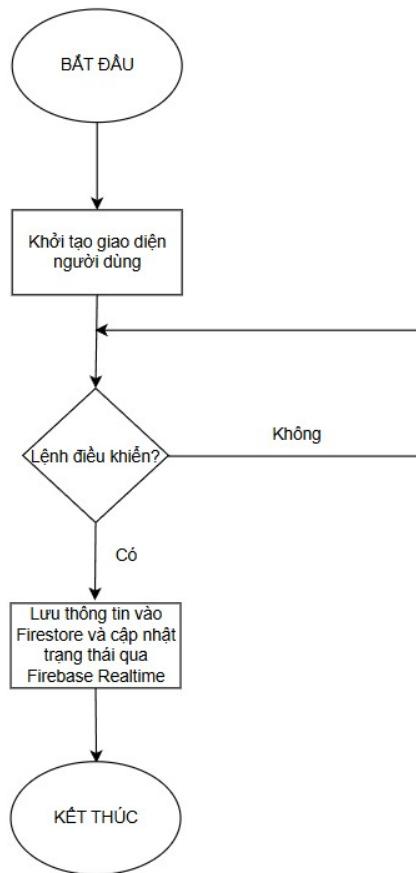


Hình 3.23: Lưu đồ thuật toán quá trình xác thực dữ liệu người dùng

Lưu đồ trên mô tả quy trình xác thực và quản lý người dùng tại tầng ứng dụng của hệ thống. Quá trình bắt đầu bằng việc kiểm tra trạng thái đăng nhập của người

dùng. Trường hợp người dùng chưa đăng nhập, hệ thống cho phép thực hiện đăng ký tài khoản và lưu trữ thông tin người dùng vào khói lưu trữ dữ liệu. Khi người dùng thực hiện đăng nhập, hệ thống tiến hành kiểm tra thông tin xác thực và đối chiếu dữ liệu trong khói lưu trữ. Nếu thông tin hợp lệ, người dùng được chuyển đến giao diện phù hợp để thực hiện các thao tác giám sát và điều khiển tiếp theo.

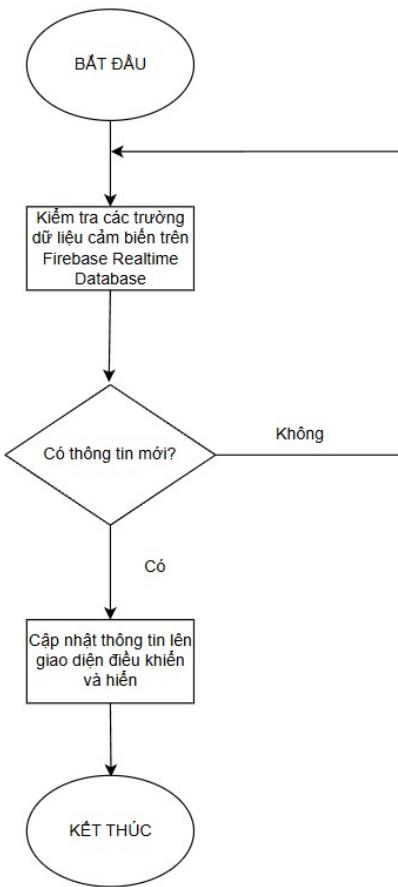
Tiếp theo là quá trình gửi dữ liệu xuống tầng gateway:



Hình 3.24: Lưu đồ thuật toán quá trình gửi dữ liệu xuống gateway

Tại giao diện người dùng với đúng quyền (Admin, User) người dùng tiến hành các thao tác điều khiển thiết bị tại hiện trường, thông tin sẽ được lưu trữ lại trong Firestore theo cú pháp ”db/username/rule/device/status” (phần này sẽ được trình bày chi tiết bên dưới khói lưu trữ) và cập nhật các trạng thái thiết bị để đồng bộ hệ thống qua việc cập nhật Firebase Realtime Database.

Cuối cùng là lưu đồ minh họa quá trình nhận dữ liệu từ tầng gateway:



Hình 3.25: Lưu đồ thuật toán quá nhận dữ liệu từ gateway

Khi có dữ liệu cảm biến từ tầng gateway gửi lên Firebase Realtime Database, tầng ứng dụng sẽ kiểm tra định kỳ và cập nhật thông tin với đúng giao diện, đúng quyền, đúng loại cảm biến và đồng thời thực hiện công việc trực quan hóa dữ liệu.

Tóm lại, ba quá trình trên là cơ sở để đi đến thiết kế chi tiết cho hai thành phần con trong tầng ứng dụng là khối lưu trữ và khối hiển thị và điều khiển.

### 3.7.3.1 Thiết kế giao diện khối hiển thị và điều khiển

Khối hiển thị và điều khiển được thiết kế nhằm cung cấp giao diện trực quan cho người dùng trong việc giám sát trạng thái hệ thống và thực hiện các thao tác điều khiển từ xa. Đồng thời, khối này phải đảm bảo mọi thao tác điều khiển đều tuân thủ cơ chế phân quyền và bảo mật, không cho phép can thiệp trực tiếp xuống các tầng bên dưới.

Cụ thể, khối hiển thị và điều khiển sẽ thực hiện các chức năng chính sau:

- Hiển thị giao diện với đúng quyền đăng ký phân phối.
- Hiển thị dữ liệu cảm biến theo thời gian thực, bao gồm: nhiệt độ, độ đục, pH, mực nước và các thông số môi trường liên quan.
- Hiển thị trạng thái hoạt động của các thiết bị chấp hành như bơm, van, đèn chiếu sáng và quạt khuấy.
- Cho phép người dùng gửi lệnh điều khiển hệ thống theo chế độ thủ công hoặc tự động.
- Tiếp nhận và hiển thị các cảnh báo khi thông số vượt ngưỡng an toàn hoặc xảy ra sự cố vận hành.

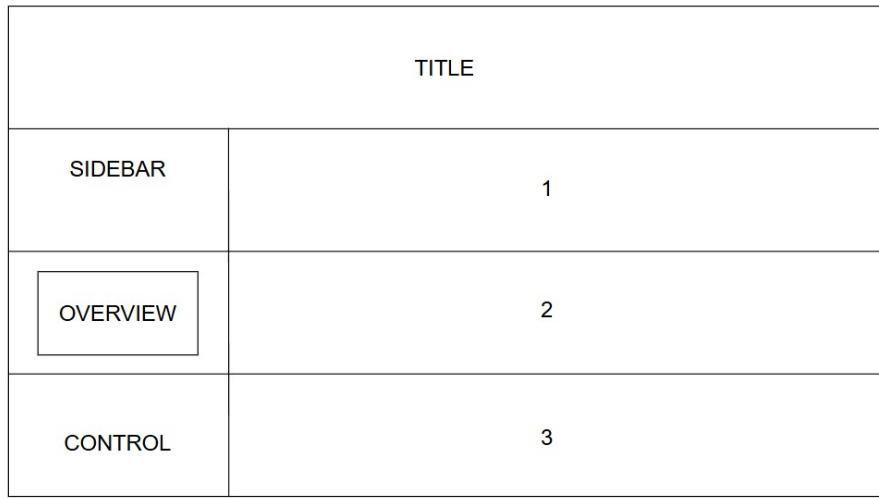
Như đã trình bày ở các phần trên thì khái niệm hiển thị và điều khiển sẽ được chia thành ba giao diện với ba quyền khác nhau bao gồm: giao diện với quyền Admin, giao diện với quyền User1, giao diện với quyền User2.

Đầu tiên ta sẽ đi thiết kế giao diện với quyền Admin: giao diện được thiết kế dành cho người quản trị hệ thống, có quyền cao nhất trong mô hình. Admin không chỉ theo dõi dữ liệu cảm biến mà còn có khả năng điều khiển các thiết bị thuộc cả hai khu vực vận hành, qua đó hỗ trợ công tác quản lý và xử lý sự cố khi cần thiết.

Về yêu cầu, giao diện Admin cần đáp ứng các yêu cầu sau:

- Hiển thị tổng quan trạng thái của toàn bộ hệ thống.
- Cho phép giám sát dữ liệu cảm biến của cả bể thu gom và bể xả.
- Cho phép điều khiển thiết bị thuộc cả hai khu vực.
- Hiển thị cảnh báo và trạng thái hệ thống một cách rõ ràng, dễ nhận biết.
- Đảm bảo mọi thao tác điều khiển đều được thực hiện thông qua tầng gateway, không giao tiếp trực tiếp với tầng thiết bị.

Định hình thiết kế tổng thể:

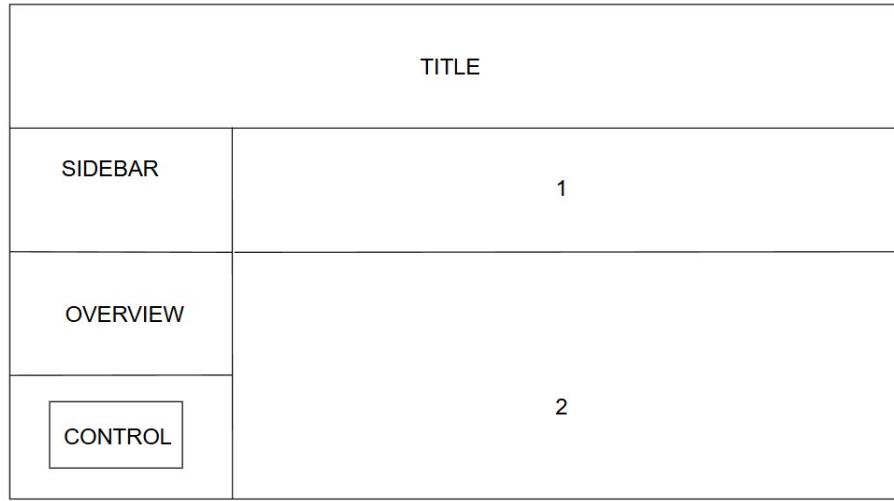


Hình 3.26: Định hình tổng quan giao diện Admin

Hình trên trình bày ý tưởng thiết kế của giao diện trong đó gồm các thành phần:

- Title (Thanh tiêu đề): Hiển thị tên hệ thống, trạng thái hoạt động và thời gian hệ thống đã hoạt động.
- Sidebar (Thanh điều hướng): Cung cấp các mục chức năng chính của hệ thống, cho phép người dùng chuyển đổi nhanh giữa các khu vực giám sát và điều khiển theo đúng quyền được phân công. Hình trên là đang ở khu vực giám sát chung (Overview).
- Khu vực nội dung 1 – Tổng quan: Hiển thị dữ liệu giám sát tổng hợp, phản ánh trạng thái chung của hệ thống xử lý nước thải.
- Khu vực nội dung 2 – Giám sát chi tiết: Trình bày chi tiết các thông số cảm biến của từng khu vực/bể xử lý, hỗ trợ người dùng theo dõi tình trạng vận hành cụ thể.
- Khu vực nội dung 3 – Thông báo: Hiển thị các thông báo hệ thống.

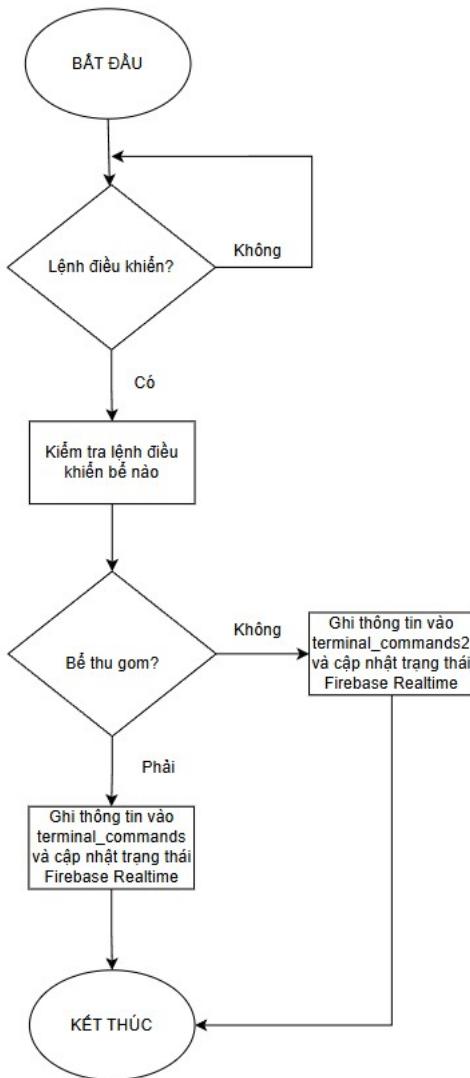
Tiếp theo là định hình thiết kế giao diện điều khiển dưới quyền Admin:



Hình 3.27: Định hình tổng quan giao diện điều khiển với quyền Admin

Một số phần vẫn giữ nguyên như ở trên, tuy nhiên có khu vực 1 sẽ là nơi để chuyển đổi qua lại để điều khiển các bệ và đồng thời điều khiển các lệnh hệ thống, khu vực 2 là nơi để điều khiển các thiết bị trong từng bệ sau khi chuyển đổi ở khu vực 1.

Khi có các lệnh điều khiển từ người dùng tương tác với giao diện quyền Admin, thì chương trình sẽ thực hiện các bước như sau:



Hình 3.28: Lưu đồ minh họa chương trình xử lý lệnh điều khiển với Admin

Mô tả: khi người dùng có lệnh thao tác điều khiển thì chương trình sẽ kiểm tra xem với quyền Admin đang điều khiển trên bể nào, nếu là bể thu gom thì sẽ lưu lệnh vào trong Firestore với trường terminal\_commands, nếu là bể xả thì sẽ lưu vào trường terminal\_commands2.

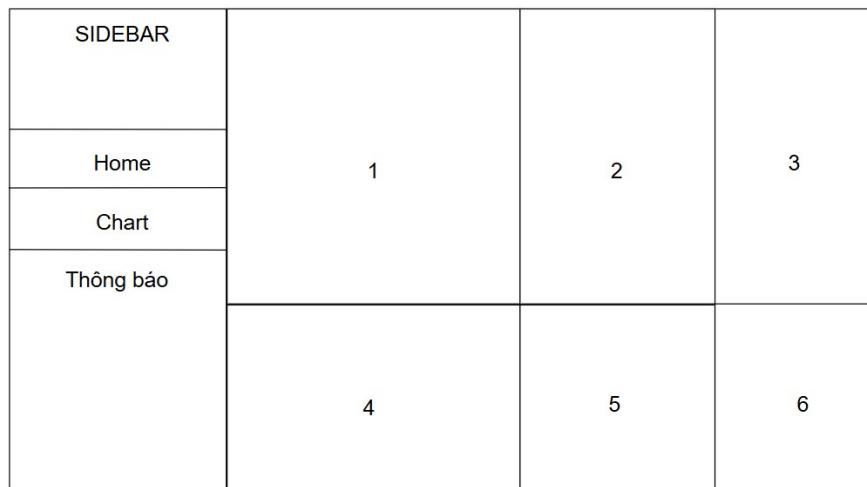
Tiếp theo là phần thiết kế giao diện với quyền User1: giao diện được thiết kế cho người vận hành tại khu vực bể thu gom nước thải. Giao diện này chỉ cung cấp các chức năng liên quan trực tiếp đến khu vực được phân công, không cho phép truy cập hoặc điều khiển các thành phần ngoài phạm vi.

Về yêu cầu, giao diện cần đảm bảo:

- Chỉ hiển thị dữ liệu cảm biến của bể thu gom.

- Chỉ cho phép điều khiển các thiết bị thuộc bể thu gom.
- Đảm bảo bảo mật dữ liệu và chỉ được phép điều khiển khi tầng gateway đã xác thực.

**Định hình thiết kế:**



Hình 3.29: Định hình tổng quan giao diện User1

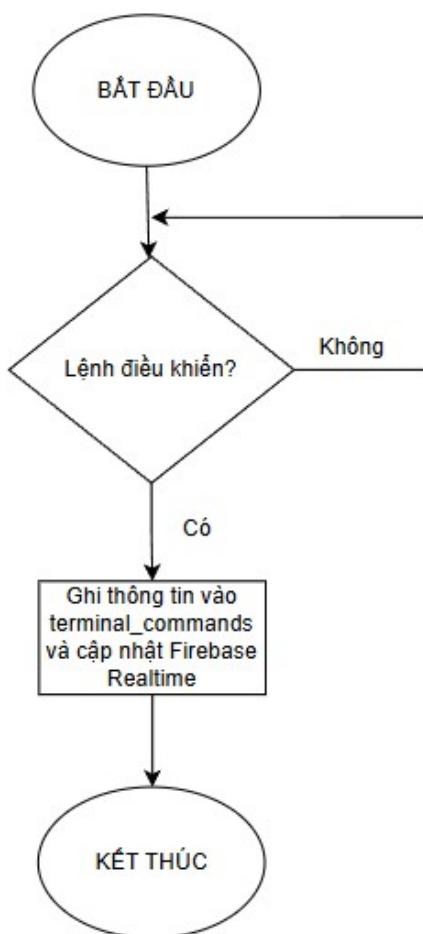
Hình trên trình bày ý tưởng thiết kế giao diện dành cho người dùng User1, tập trung vào chức năng giám sát và điều khiển cục bộ theo phạm vi được phân quyền. Giao diện được bố trí theo dạng bảng điều khiển (dashboard), trong đó bao gồm các thành phần chính như sau:

- Sidebar (Thanh điều hướng): Hiển thị thông tin người dùng đang đăng nhập và cung cấp các mục chức năng cơ bản như trang tổng quan, biểu đồ dữ liệu và khu vực thông báo.
- Khu vực nội dung 1 – Thông tin điều khiển: Hiển thị thông tin các lệnh đã được gửi bởi người dùng quyền User1.
- Khu vực nội dung 2 – Khu vực các nút điều khiển: Tại đây cho phép người dùng điều khiển các thiết bị trong quy trình của mình với đúng quyền.
- Khu vực nội dung 3 – Hiển thị mực nước.

- Khu vực nội dung 4 – Chế độ vận hành: Cho phép chuyển đổi giữa các chế độ điều khiển như dừng khẩn cấp hoặc chế độ tự động.
- Khu vực nội dung 5 – Hiển thị nhiệt độ: Trình bày giá trị nhiệt độ đo được từ cảm biến dưới dạng số.
- Khu vực nội dung 6 – Hiển thị độ đục.

Giao diện User1 được thiết kế theo hướng trực quan, đơn giản và tập trung vào giám sát – điều khiển trong phạm vi cho phép, đảm bảo người dùng có thể vận hành hệ thống hiệu quả mà không can thiệp vào các chức năng quản trị cấp cao.

Dưới đây là lưu đồ mô tả quá trình điều khiển của người dùng trên giao diện User1:



Hình 3.30: Lưu đồ mô tả quá trình điều khiển của người dùng quyền User1

Ở giao diện User1, khi người dùng phát sinh lệnh điều khiển, mọi thông tin sẽ

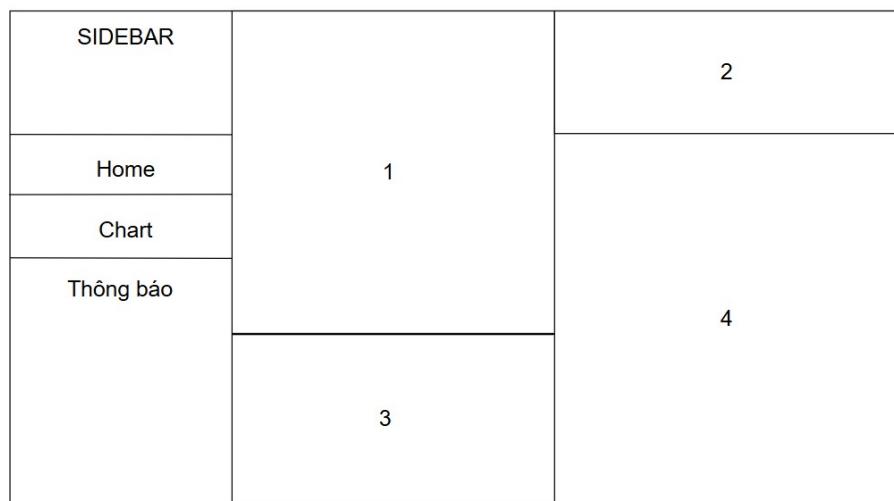
được lưu trữ vào terminal\_commands bên Firestore và đồng thời cập nhật trạng thái qua Firebase Realtime Database để đồng bộ trạng thái trên giao diện.

Cuối cùng, thiết kế giao diện với quyền User2: giao diện User2 được thiết kế cho người vận hành tại khu vực bể xã nước thải sau xử lý. Tương tự User1, giao diện này chỉ phục vụ cho một khu vực cụ thể, đảm bảo nguyên tắc phân quyền và tránh nhầm lẫn trong quá trình vận hành.

Giao diện User2 cần đáp ứng:

- Hiển thị dữ liệu cảm biến của bể xã.
- Cho phép điều khiển các thiết bị thuộc bể xã.
- Đảm bảo bảo mật dữ liệu và chỉ được phép điều khiển khi tầng gateway đã xác thực.

Định hình thiết kế:



Hình 3.31: Định hình tổng quan giao diện User2

Hình trên thể hiện định hình thiết kế giao diện dành cho người dùng User2, với các thành phần chính của giao diện bao gồm:

- Sidebar (Thanh điều hướng): Hiển thị thông tin người dùng và các mục chức năng cơ bản như trang chủ, biểu đồ và thông báo.

- Khu vực nội dung 1 – Thông tin điều khiển: Hiển thị thông tin các lệnh đã được gửi bởi người dùng quyền User2.
- Khu vực nội dung 2 – Điều khiển thiết bị: Cung cấp các nút điều khiển trực quan cho các thiết bị chấp hành thuộc phạm vi quản lý của User2, đảm bảo thao tác điều khiển đơn giản và rõ ràng.
- Khu vực nội dung 3 – Chế độ vận hành: Cho phép chuyển đổi giữa các chế độ điều khiển như dừng khẩn cấp hoặc chế độ tự động.
- Khu vực nội dung 4 – Giám sát thông số nước: Hiển thị các thông số quan trọng như pH, độ đục, nhiệt độ và mực nước, kết hợp biểu diễn trực quan nhằm giúp người dùng dễ dàng đánh giá chất lượng nước trong bể xử lý.

Giao diện User2 được thiết kế theo hướng tối giản, trực quan và tập trung vào chức năng vận hành, đảm bảo người dùng có thể giám sát và điều khiển hiệu quả trong phạm vi được phân quyền.

Khi có lệnh điều khiển phát sinh từ người dùng, chương trình sẽ xử lý như sau:



Hình 3.32: Lưu đồ mô tả quá trình điều khiển của người dùng quyền User2

Tương tự quá trình xử lý ở User1, tuy nhiên ở đây thông tin sẽ được lưu trữ vào terminal\_commands2 của Firestore và bên Firebase Realtime Database các trạng thái sẽ được cập nhật trong User2.

### 3.7.3.2 Thiết kế khôi lưu trữ

Khôi lưu trữ dữ liệu giữ vai trò trung tâm trong tầng ứng dụng, có nhiệm vụ tiếp nhận, lưu trữ và quản lý toàn bộ dữ liệu phát sinh trong quá trình vận hành hệ thống, bao gồm dữ liệu cảm biến, trạng thái thiết bị và thông tin người dùng.

Trong kiến trúc đề xuất, khôi lưu trữ được xây dựng dựa theo mô hình lưu trữ đám mây, khôi này sẽ khai thác triệt để các ứng dụng của Firebase, cụ thể:

- Firebase Realtime Database được sử dụng để lưu trữ dữ liệu cảm biến và trạng thái thiết bị theo thời gian thực. Cơ sở dữ liệu này cho phép cập nhật và đồng bộ dữ liệu liên tục giữa gateway và tầng ứng dụng, đáp ứng yêu cầu giám sát tức thời các thông số vận hành của hệ thống.
- Cloud Firestore được sử dụng để lưu trữ các dữ liệu có cấu trúc và ít thay đổi, bao gồm thông tin tài khoản người dùng, phân quyền truy cập, lịch sử thao tác điều khiển và các sự kiện vận hành của hệ thống.
- Firebase Authentication đảm nhiệm chức năng xác thực người dùng, quản lý đăng ký và đăng nhập.

Cấu trúc lưu trữ trong Firebase:

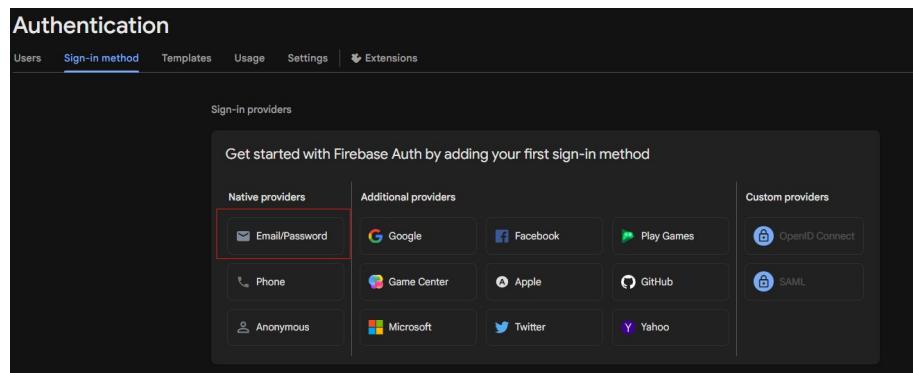
Đầu tiên, Firebase Authentication, các bước cấu hình chi tiết:

Tạo dự án mới trong Firebase:

- Truy cập <https://console.firebaseio.google.com>
- Click "Add project" hoặc "Create a project"
- Đặt tên cho project
- Sau đó đổi khu vực lại thành Taiwan

- Cuối cùng nhấn Next để tiếp tục

Tiếp theo vào dự án vừa tạo, nhấn Build tại Sidebar và chọn Authentication. Sau đó nhấn Getting Started và hiện ra bảng chọn như sau và ta sẽ chọn Email/Password như trong ảnh:



Hình 3.33: Quá trình cấu hình Firebase Authentication hình 1

Kết thúc Enable hết lên và nhấn Save, sau khi tạo xong qua lại Users sẽ xuất hiện như thế này:

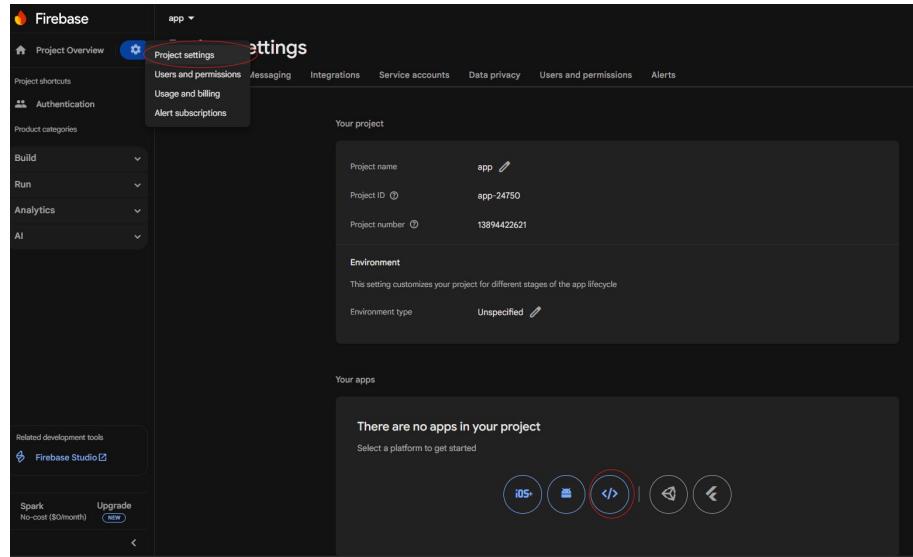
Identifier	Providers	Created	Signed In	User UID	Add user	⋮
No users for this project yet						

Hình 3.34: Cấu trúc của một Firebase Authentication

Trong ảnh có:

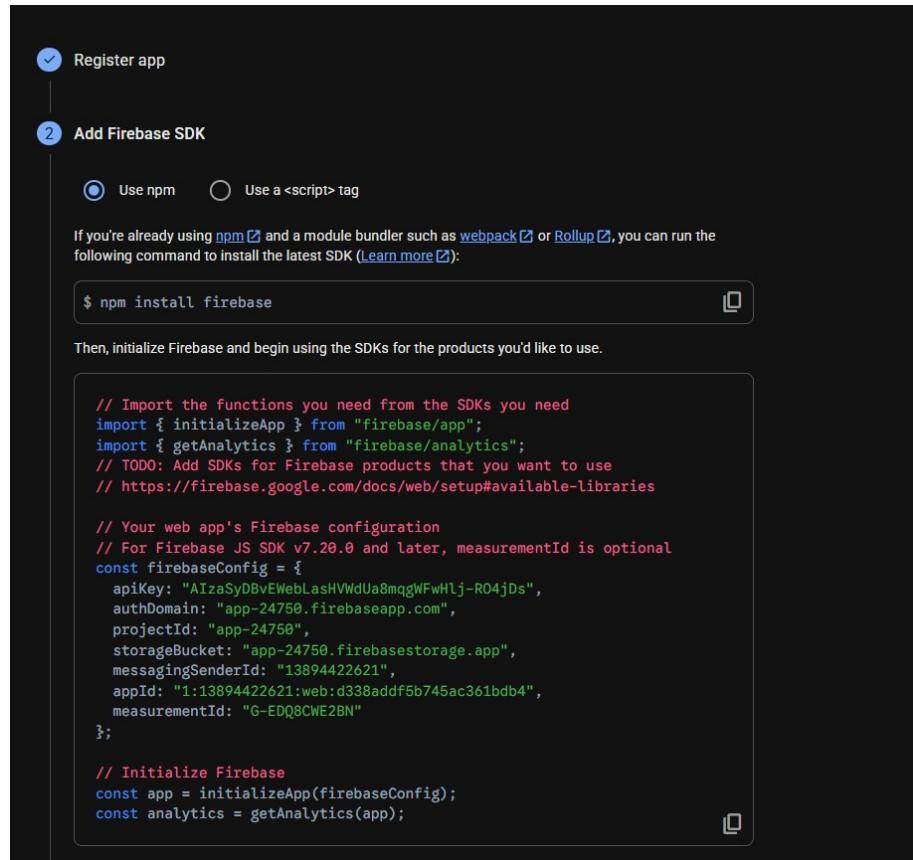
- Identifier: email đăng nhập
- Provider: phương thức xác thực
- Created: thời gian tạo
- Signed in: lần đăng nhập cuối
- User UID: ID duy nhất

Tiếp đó, chõ Project Overview có biểu tượng bánh răng, ta thực hiện nhấn vào đó chọn Project Setting và chọn Web.



Hình 3.35: Quá trình cấu hình Firebase Authentication hình 2

Sau khi nhấn vào Web, ta sẽ được chuyển đến phần tiếp theo là đặt tên và lấy cấu hình firebaseconfig, ta copy và thực hiện các bước như trong ảnh rồi nhấn hoàn thành.



Hình 3.36: Quá trình cấu hình Firebase Authentication hình 3

Như vậy thực hiện các bước trên là đã hoàn thành công việc đầu tiên chính là cấu hình cho dự án Firebase nói chung và Firebase Authentication nói riêng.

Thứ hai, đến bước cài đặt Firebase Realtime Database chi tiết sẽ được trình bày dưới đây:

Bước đầu tiên:

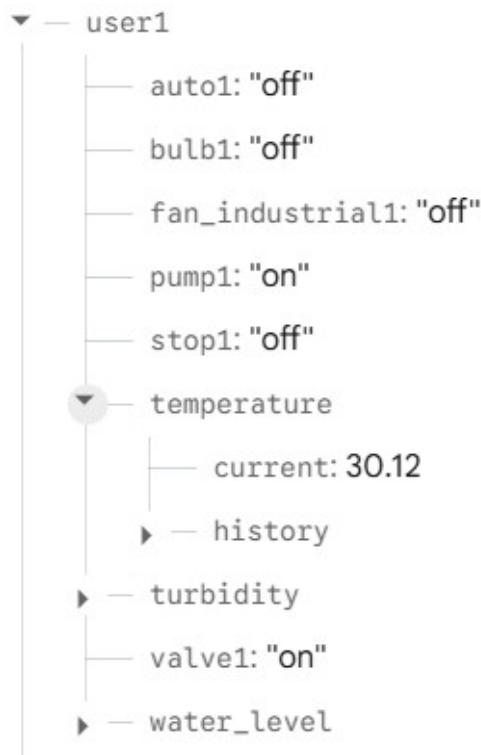
- Tại Sidebar bên trái chọn Build, chọn Realtime Database
- Click Create Database
- Chọn vị trí server chọn Singapore
- Click Enable

Tiếp theo là cấu hình Rules:

- Vào Rules

- Đổi read và write lại thành true
- Enabled

Sau khi tiến hành thêm các trường mong muốn thì bên dưới đây là cấu trúc của dự án được dùng hiện tại. Do cấu hình của user1 và user2 tương tự nhau nên ở đây chỉ lấy điển hình của user1 để trình bày:



Hình 3.37: Cấu trúc của một Firebase Realtime Database

Trong đó:

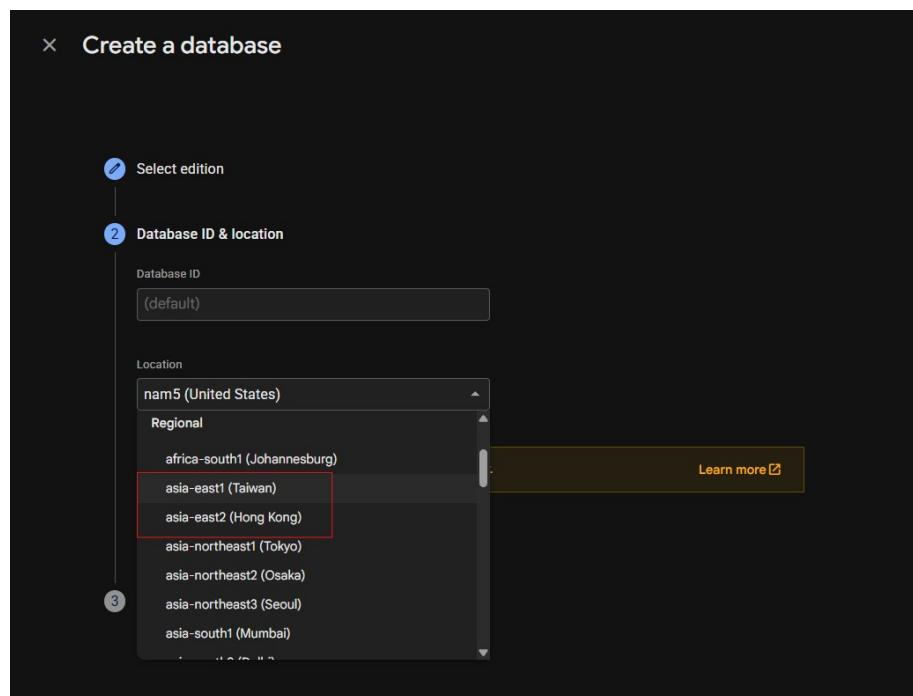
- user1: Người dùng/Thiết bị
  - bulb1, fan\_industrial1, pump1: Trạng thái thiết bị (on/off)
  - auto1, stop1: Chế độ đặc biệt của hệ thống
  - temperature:
    - + current: Nhiệt độ hiện tại
    - + history: Lịch sử ghi nhận

- turbidity: Độ đục nước
- water\_level: Mực nước

Cuối cùng, Cloud Firestore, dưới đây là cách cấu hình chi tiết:

Đầu tiên ta chọn Build, chọn Firestore Database, sau đó nhấn Create Database.

Tiếp theo sẽ đổi khu vực thành một trong hai cái như trong ảnh:



Hình 3.38: Cấu hình Firetore hình 1

Sau đó nhấn Next rồi Create để kết thúc quá trình tạo.

Kế đến là tạo Collection:

- Click Start collection
- Nhập Collection ID
- Click Next
- Tạo Document (điền tên các trường mong muốn vào)
- Click Save

Trong án này Firestore được tổ chức thành 3 collection users (lưu thông tin người dùng phục vụ cho gateway xác thực), terminal\_commands lưu thông tin các lệnh điều khiển cứng cho gateway xử lý và terminal\_commands\_2 tương tự terminal\_commands, hai collection này dùng cho user1 và user2.

Dưới đây là cấu trúc từng phần chi tiết:

```
command: "db/user2/admin/pump2/on"
processed: true
processedAt: December 25, 2025 at 3:45:48 AM UTC+7
role: "user2"
type: "input"
```

Hình 3.39: Cấu trúc của collection terminal\_commands\_2

Trong đó:

- command: "db/user2/admin/pump2/on" - Đường dẫn lệnh trong database để điều khiển pump2
- processed: true - Trạng thái xử lý lệnh (đã xử lý hay chưa)
- processedAt: Thời gian lệnh được xử lý
- role: "user2" - Vai trò của người thực hiện lệnh
- type: "input" - Loại lệnh

```
account: "thai_admin"  
createdAt: "17:11:11 16/12/2025"  
email: "thai_admin@myapp.local"  
password: "123456"  
processed: true  
processedAt: "2025-12-16T10:11:11.749565"  
role: "admin"
```

Hình 3.40: Cấu trúc của collection users

Trong đó:

- account: Tên tài khoản người dùng
- createdAt: Thời gian tạo tài khoản
- email: Email giả dùng cho Firebase Authentication
- password: Mật khẩu
- processed: true - Trạng thái đã được gateway xử lý
- processedAt: Thời gian xử lý/kích hoạt tài khoản
- role: Vai trò/quyền hạn của tài khoản

#### 3.7.4 Thiết kế phần mềm tầng gateway

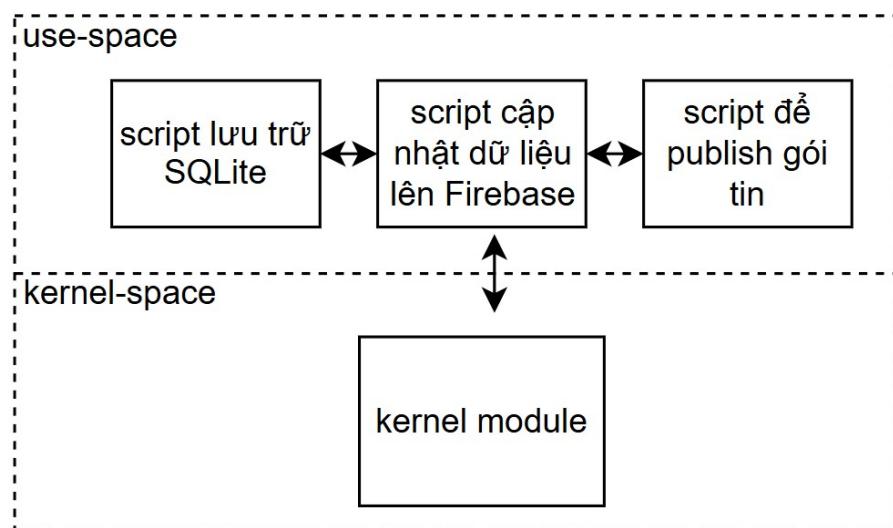
Tầng gateway đóng vai trò trung tâm trong kiến trúc phần mềm của hệ thống IoT, là lớp trung gian kết nối giữa tầng thiết bị hiện trường và tầng ứng dụng. Phần mềm tại gateway chịu trách nhiệm tiếp nhận dữ liệu từ các node thiết bị, xử lý – kiểm tra tính hợp lệ, đồng thời điều phối luồng dữ liệu và lệnh điều khiển giữa các thành phần trong hệ thống. Việc thiết kế phần mềm gateway hợp lý giúp đảm bảo hệ thống vận hành ổn định, an toàn và có khả năng mở rộng.

Phần mềm gateway không trực tiếp giao tiếp với người dùng cuối, mà hoạt động như một bộ điều phối trung tâm với các nhiệm vụ chính:

- Tiếp nhận dữ liệu cảm biến từ tầng thiết bị thông qua giao thức MQTT.
- Kiểm tra, xác thực và lọc dữ liệu trước khi chuyển lên tầng lưu trữ.
- Tiếp nhận lệnh điều khiển từ tầng ứng dụng, kiểm tra quyền truy cập và tính hợp lệ của lệnh.
- Chuyển tiếp lệnh điều khiển xuống đúng thiết bị tương ứng.

Nhờ đó, gateway giúp tách biệt hoàn toàn tầng thiết bị khỏi Internet và giao diện người dùng, giảm thiểu nguy cơ tấn công trực tiếp vào hệ thống điều khiển.

Phần mềm gateway được triển khai trên nền hệ điều hành Linux (Raspberry Pi OS) và được tổ chức theo mô hình dưới đây:



Hình 3.41: Kiến trúc phần mềm trên gateway

Trong đó:

- Script lưu trữ SQLite: chịu trách nhiệm thu thập và lưu trữ dữ liệu cảm người dùng khi đăng ký vào cơ sở dữ liệu SQLite cục bộ trên Raspberry Pi. Và trả về kết quả để xác thực khi cần.

- Script cập nhật dữ liệu lên Firebase: chuyển dữ liệu cảm biến từ tầng thiết bị lên Firebase Realtime Database.
- Script để publish gói tin: kiểm tra gói tin điều khiển theo đúng qui trình rồi mới chuyển cho tầng thiết bị thông qua MQTT.
- Kernel module: module lọc gói tin mạng (firewall) hoạt động tại hook point NF\_INET\_LOCAL\_IN. Module này kiểm tra và lọc tất cả traffic đến, chỉ cho phép các port cần thiết (22-SSH, 53-DNS, 443-HTTPS, 1883-MQTT, 67/68-DHCP). Điều này bảo vệ Raspberry Pi khỏi các truy cập trái phép, đảm bảo an toàn cho hệ thống IoT.

Cơ chế hoạt động:

Gateway hoạt động trong mô hình được thiết kế chia làm hai cơ chế: cơ chế xử lý dữ liệu cảm biến và cơ chế xử lý dữ liệu điều khiển.

Thứ nhất, cơ chế xử lý dữ liệu cảm biến:

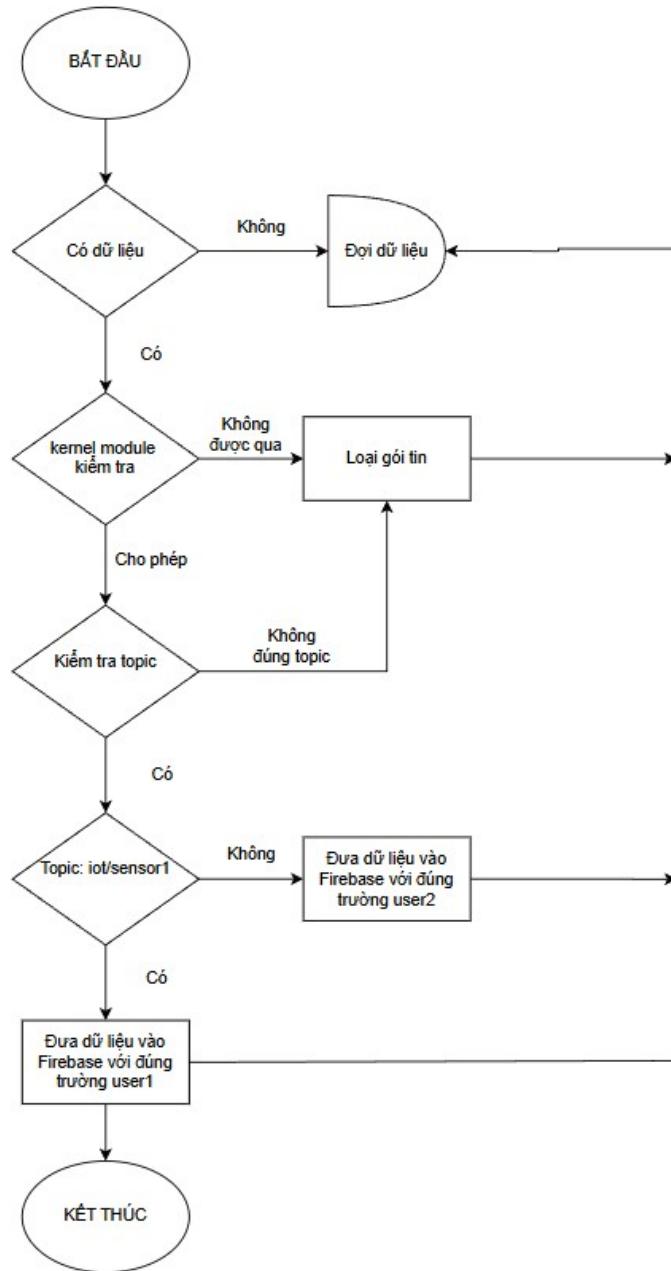
Dữ liệu cảm biến từ tầng thiết bị được gửi về gateway theo chu kỳ hoặc theo sự kiện thông qua các topic MQTT đã định nghĩa. Khi nhận dữ liệu, gateway thực hiện:

- Kiểm tra nguồn gửi và topic hợp lệ.
- Phân loại dữ liệu theo từng bể và từng nhóm cảm biến.
- Ghi dữ liệu thời gian thực lên Firebase Realtime Database để phục vụ giám sát.

Thứ hai, cơ chế xử lý dữ liệu điều khiển:

- Kiểm tra trạng thái xử lý của lệnh (chưa xử lý/đã xử lý).
- Xác thực quyền điều khiển dựa trên vai trò người dùng (Admin, User1, User2).
- Chuyển đổi lệnh điều khiển thành định dạng phù hợp để gửi xuống thiết bị.
- Cập nhật trạng thái thực thi và phản hồi về tầng ứng dụng.

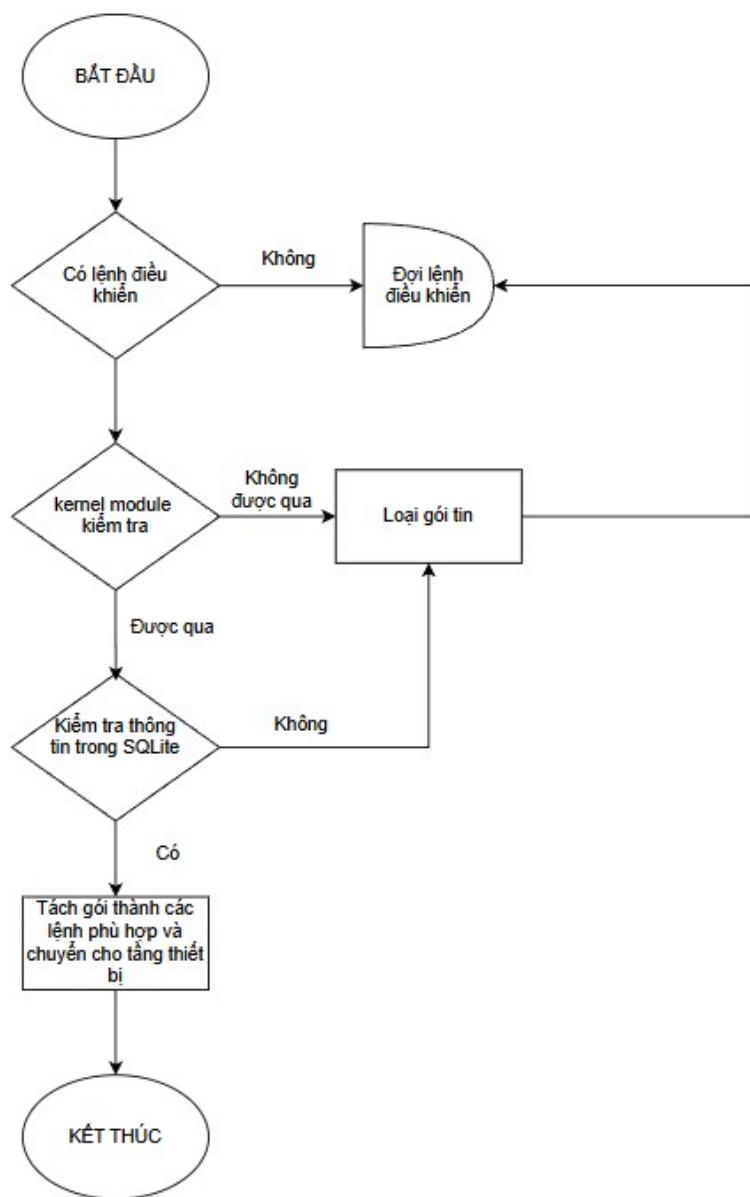
Dưới đây là lưu đồ trình bày luồng hoạt động của cơ chế xử lý dữ liệu cảm biến:



Hình 3.42: Lưu đồ cơ chế xử lý dữ liệu cảm biến

Lưu đồ trên với mục đích mô tả chương trình đưa dữ liệu cảm biến từ tầng thiết bị lên tầng ứng dụng thông qua tầng gateway, với cơ chế bảo mật nghiêm ngặt, kiểm tra từng bước từ kernel-space đến user-space.

Tiếp theo là lưu đồ cơ chế xử lý dữ liệu điều khiển trên gateway:



Hình 3.43: Lưu đồ cơ chế xử lý dữ liệu điều khiển

Qua quá trình thiết kế phần mềm tầng gateway, có thể thấy gateway đóng vai trò trung tâm trong việc kết nối, kiểm soát và điều phối toàn bộ hoạt động của hệ thống IoT. Gateway không chỉ thực hiện chức năng trung chuyển dữ liệu giữa tầng thiết bị và tầng ứng dụng, mà còn là lớp kiểm soát quan trọng nhằm đảm bảo an toàn và tính toàn vẹn của hệ thống.

### 3.7.5 Thiết kế phần mềm tầng thiết bị

Tầng thiết bị là tầng thấp nhất trong kiến trúc hệ thống IoT, trực tiếp kết nối với môi trường vật lý thông qua các cảm biến và cơ cấu chấp hành. Trong mô hình đề xuất, tầng thiết bị không tham gia vào quá trình xác thực người dùng, không lưu trữ thông tin nhạy cảm và không cho phép kết nối trực tiếp với Internet. Mọi hoạt động giao tiếp ra bên ngoài đều phải thông qua gateway, từ đó đảm bảo thiết bị không trở thành điểm tấn công trực tiếp từ mạng công cộng.

Phần mềm tầng thiết bị được xây dựng trên kiến trúc hai vi điều khiển, bao gồm ESP32 và STM32, mỗi vi điều khiển đảm nhiệm một vai trò riêng biệt:

- ESP32 đảm nhiệm chức năng truyền thông mạng, đóng vai trò trung gian giữa tầng thiết bị và gateway.
- STM32 đảm nhiệm chức năng xử lý tại hiện trường, bao gồm đọc cảm biến và điều khiển các cơ cấu chấp hành.

Thiết kế phần mềm trên ESP32:

Phần mềm trên ESP32 sẽ thực hiện các chức năng chính sau:

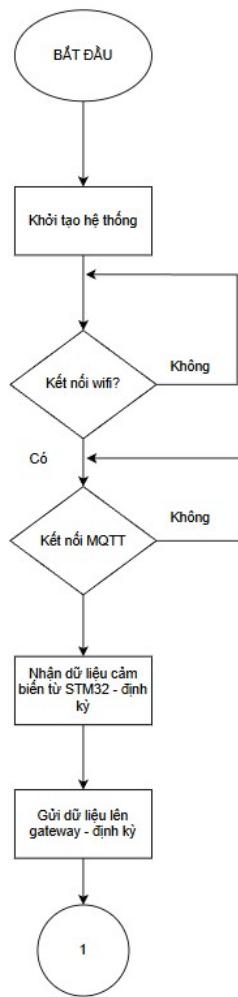
- Kết nối mạng Wi-Fi với thông tin được cấu hình trước.
- Kết nối tới MQTT Broker tại gateway thông qua địa chỉ IP và cổng xác định (1883).
- Nhận lệnh điều khiển hợp lệ từ gateway.
- Gửi dữ liệu cảm biến do STM32 cung cấp lên gateway.
- ESP32 không trực tiếp xử lý logic điều khiển chi tiết mà chỉ đóng vai trò trung gian truyền – nhận dữ liệu có kiểm soát.

Để giảm thiểu nguy cơ bị khai thác trái phép, ESP32 được cấu hình các ràng buộc bảo mật sau:

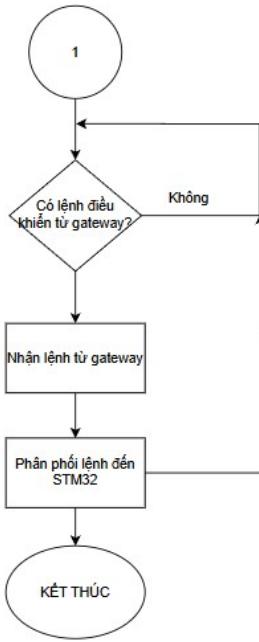
- Chỉ cho phép kết nối duy nhất một MQTT Broker đã được định nghĩa trước.

- Chỉ subscribe một topic điều khiển cố định.
- Chỉ publish dữ liệu cảm biến lên các topic được chỉ định.
- Không hỗ trợ các giao thức khác như HTTP hay WebSocket.
- Không nhận lệnh điều khiển trực tiếp từ giao diện người dùng hoặc Internet.

Cụ thể hơn, lưu đồ dưới đây sẽ trình bày cách ESP32 hoạt động:



Hình 3.44: Lưu đồ thuật toán trên ESP32



Hình 3.45: Lưu đồ thuật toán trên ESP32 tiếp theo

Trình tự hoạt động tổng quát của ESP32 bắt đầu từ quá trình khởi động hệ thống và thực hiện kết nối vào mạng Wi-Fi đã được cấu hình. Sau khi kết nối mạng thành công, ESP32 tiến hành thiết lập kết nối tới MQTT Broker để sẵn sàng cho việc trao đổi dữ liệu. Trong trường hợp kết nối MQTT chưa được thiết lập hoặc bị gián đoạn, thiết bị sẽ tạm thời không thực hiện gửi hoặc nhận dữ liệu nhằm đảm bảo tính ổn định và an toàn của hệ thống. Khi nhận được lệnh điều khiển hợp lệ từ gateway thông qua kênh truyền MQTT, ESP32 sẽ chuyển tiếp lệnh này xuống vi điều khiển STM32 thông qua giao tiếp UART để thực hiện điều khiển cơ cấu chấp hành. Song song với đó, ESP32 thực hiện gửi dữ liệu cảm biến lên gateway theo chu kỳ định sẵn, phục vụ cho quá trình giám sát và xử lý ở các tầng phía trên.

Tiếp theo là thiết kế phần mềm trên STM32:

Phần mềm trên vi điều khiển STM32 được thiết kế dựa trên hệ điều hành thời gian thực FreeRTOS, nhằm đảm bảo khả năng xử lý song song, đáp ứng thời gian thực và vận hành ổn định khi hệ thống mở rộng số lượng cảm biến và thiết bị chấp hành. STM32 đóng vai trò là bộ xử lý hiện trường, trực tiếp thu thập dữ liệu cảm biến và thực thi lệnh điều khiển, không tham gia vào các cơ chế xác thực hay giao

tiếp mạng.

Phần mềm STM32 được tổ chức theo mô hình đa nhiệm, các task bao gồm:

- Task đọc cảm biến 1: Thực hiện đọc dữ liệu từ các cảm biến như nhiệt độ, độ đục và mực nước theo chu kỳ định sẵn.
- Task cảm biến 2: Thực hiện đọc dữ liệu từ các cảm biến như nhiệt độ, độ đục, mực nước và pH theo chu kỳ định sẵn.

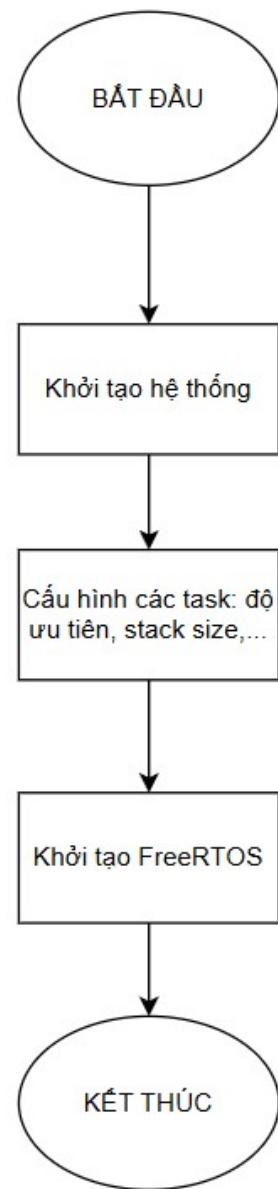
Ngoài ra STM32 sử dụng các ngắt phần cứng cho các sự kiện quan trọng như: Nhận dữ liệu UART từ ESP32, kết thúc quá trình chuyển đổi ADC, sự kiện thời gian từ timer.

STM32 sử dụng hai kênh giao tiếp UART độc lập nhằm tách biệt luồng dữ liệu cảm biến và luồng dữ liệu điều khiển. Cụ thể, UART1 được sử dụng để truyền dữ liệu cảm biến từ STM32 lên ESP32 theo chu kỳ định sẵn, bao gồm các thông số nhiệt độ, độ đục, mực nước và pH. Trong khi đó, UART2 được sử dụng riêng cho việc tiếp nhận và gửi các lệnh điều khiển từ ESP32 xuống STM32.

Do nhiều task có thể cùng truy cập vào các tài nguyên dùng chung như ADC, UART hoặc vùng dữ liệu cảm biến, hệ thống sử dụng mutex của FreeRTOS để đảm bảo tính toàn vẹn dữ liệu và tránh xung đột truy cập. Cơ chế này giúp cho việc:

- Ngăn chặn việc đọc/ghi dữ liệu đồng thời gây sai lệch kết quả.
- Đảm bảo thứ tự xử lý rõ ràng giữa các task.
- Tăng độ ổn định khi hệ thống hoạt động liên tục trong thời gian dài.

Dưới đây sẽ trình bày lưu đồ chương trình cấu hình ban đầu trong hàm main:

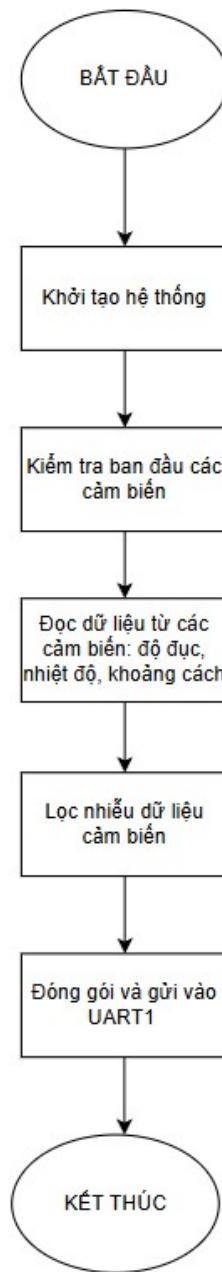


Hình 3.46: Lưu đồ cấu hình hàm main

Trong đó: Khởi tạo hệ thống sẽ chạy các hàm init của: ADC (dùng cho cảm biến độ đục, cảm biến pH), Timer (dùng cho cảm biến nhiệt độ, cảm biến khoảng cách), các định nghĩa GPIO,...

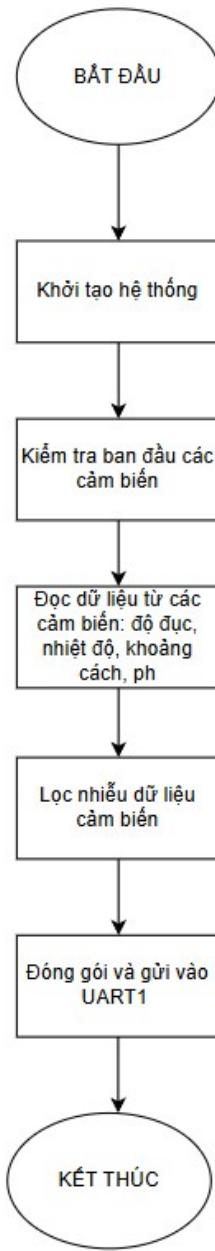
Tiếp theo sẽ trình bày lưu đồ thuật toán dùng trong task cảm biến 1 và task cảm biến 2.

Lưu đồ cho task cảm biến 1:



Hình 3.47: Lưu đồ thuật toán dùng cho task cảm biến 1

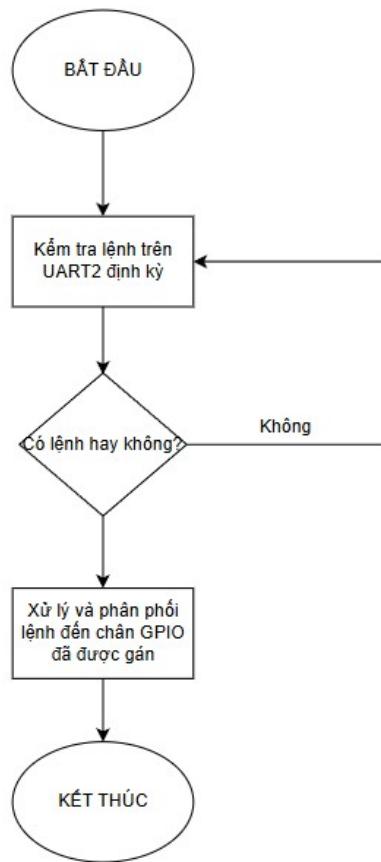
Lưu đồ thuật toán trên trình bày quy trình hoạt động của task cảm biến 1, cụ thể chương trình sẽ đọc các cảm biến khoảng cách, cảm biến nhiệt độ, cảm biến độ đục để lấy giá trị, các chương trình đọc cảm biến phải đảm bảo nằm trong mutex để giới hạn lại thứ tự trước sau và đảm bảo phải áp dụng các thuật toán lọc nhiễu cảm biến. Cuối cùng là đóng gói lại và gửi vào UART1 để chuyển cho ESP32.



Hình 3.48: Lưu đồ thuật toán dùng cho task cảm biến 2

Lưu đồ mô tả quá trình thu thập và truyền dữ liệu của task cảm biến 2. Sau khi khởi tạo hệ thống, chương trình kiểm tra kết nối các cảm biến, đọc dữ liệu (độ đục, nhiệt độ, mực nước, pH), lọc nhiễu để tăng độ chính xác, sau đó đóng gói và truyền dữ liệu qua giao tiếp UART1.

Cuối cùng là lưu đồ nhận dữ liệu và thực thi lệnh điều khiển trên UART2:



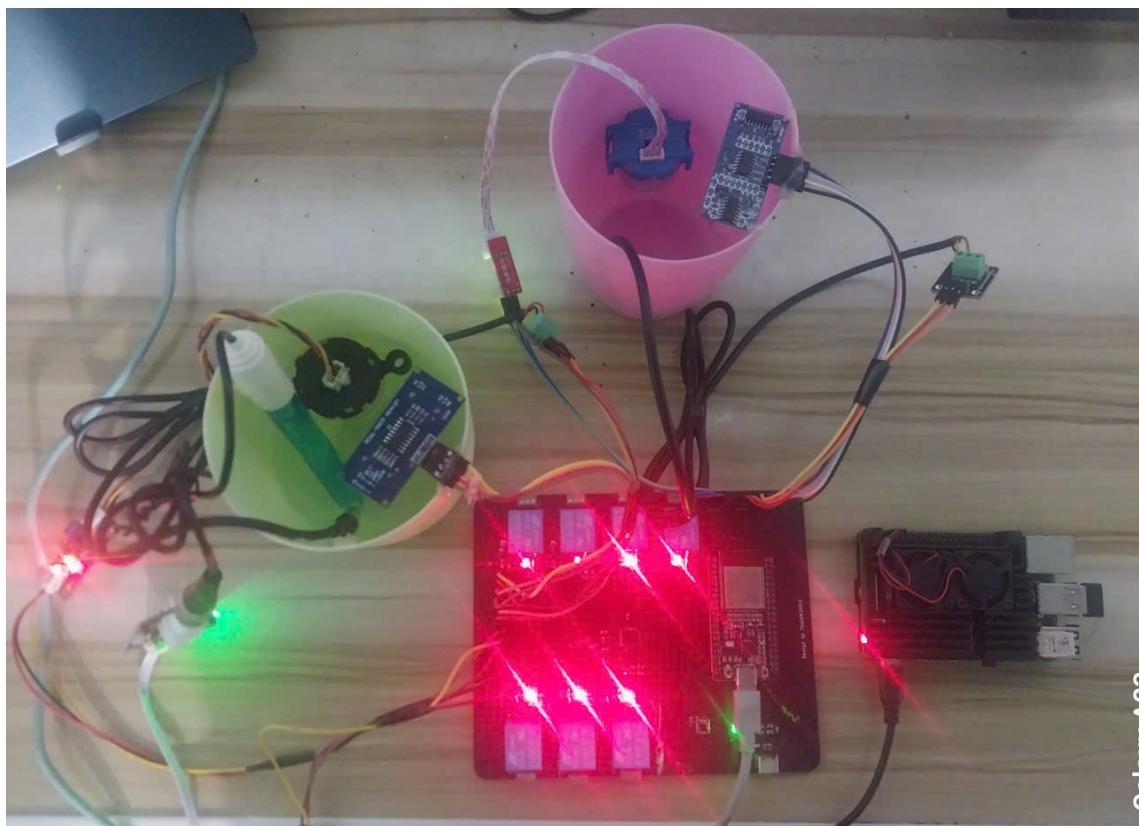
Hình 3.49: Lưu đồ thuật toán điều phối lệnh điều khiển trên UART2

Khi có lệnh gửi về thì chương trình sẽ có ngắt UART được thực hiện trên UART2, tại đây vi điều khiển sẽ kiểm tra thông tin về thiết bị và trạng thái của thiết bị. Sau đó sẽ điều phối lệnh đó đến chân GPIO đã được định nghĩa trước trong chương trình.

## CHƯƠNG 4. KẾT QUẢ VÀ ĐÁNH GIÁ

Sau quá trình thiết kế, thi công và kiểm thử, hệ thống mô hình IoT bảo mật đã tầng ứng dụng trong quy trình xử lý nước thải công nghiệp đã được hoàn thiện và vận hành ổn định. Chương này trình bày các kết quả thu được từ quá trình triển khai phần cứng và phần mềm, đồng thời đánh giá mức độ đáp ứng của hệ thống so với các yêu cầu kỹ thuật đã đề ra.

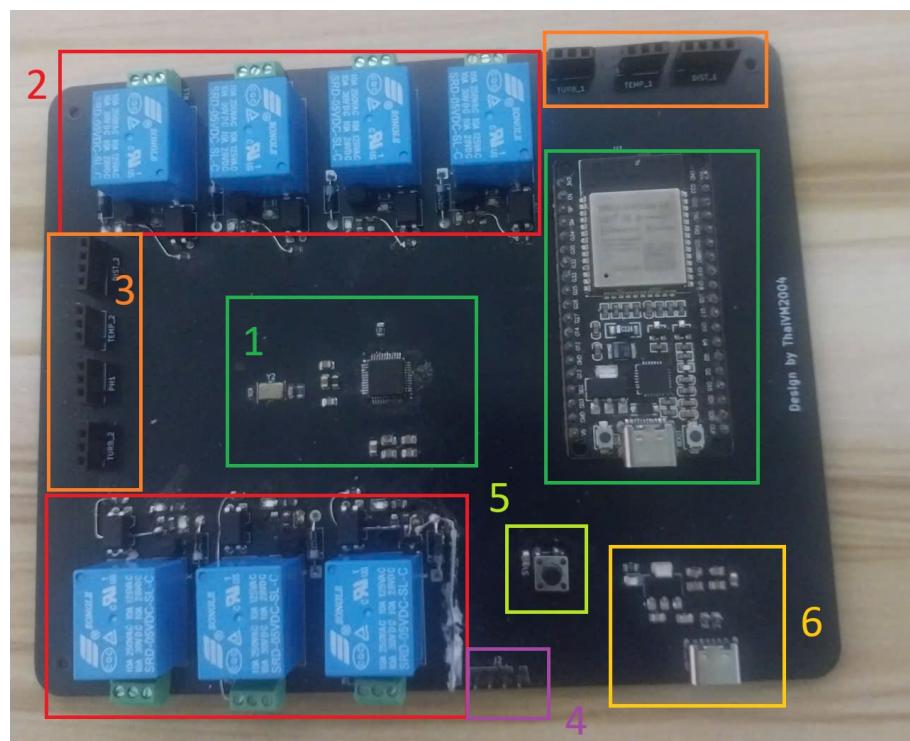
Trong phần cứng, các khối chức năng chính bao gồm khối vi điều khiển, khối cảm biến, khối điều khiển cơ cấu chấp hành, khối gateway và khối nguồn đã được tích hợp thành một hệ thống hoàn chỉnh. Hình ảnh dưới đây minh họa mô hình thực tế sau khi lắp ráp, kết nối và cấp nguồn cho toàn bộ hệ thống:



Hình 4.1: Mô hình đã hoàn thiện

## 4.1 KẾT QUẢ PHẦN CỨNG

Hệ thống phần cứng của mô hình IoT bảo mật đa tầng đã được lắp ráp hoàn chỉnh và kiểm thử thành công. Mô hình gồm nhiều khối chức năng được tích hợp chặt chẽ, đảm bảo khả năng giám sát và điều khiển theo thời gian thực, đồng thời đáp ứng các yêu cầu về bảo mật và phân quyền. Dưới đây sẽ trình bày chi tiết kết quả thiết kế trên tầng thiết bị:



Hình 4.2: Kết quả thiết kế tầng thiết bị

Trong đó:

- 1 là khối vi điều khiển gồm hai vi điều khiển là STM32F103c8T6 và ESP32 Wrom32 Module. Hai vi điều khiển này được đặt cạnh nhau nhất có thể để đảm bảo đường đi của giao thức UART được truyền ngắn nhất. Bên cạnh vi điều khiển STM32 có thạch anh để cấp nguồn dao động cho vi điều khiển và các tụ lọc đi kèm để đảm bảo nguồn không bị sụt áp trong quá trình truyền đi xa.
- 2 là khối điều khiển cơ cầu chìp hành, khối này dùng các relay 5v để mô

phóng hoạt động của các thiết bị trong công nghiệp thực tế, trong dự án này dùng 7 relay để mô phỏng cho 7 thiết bị. Khối điều khiển cơ cấu chấp hành được đặt gần outline của board để đảm bảo an toàn cho khối vi điều khiển vì khi kích relay có thể gây ra điện áp ngược, từ đó làm ảnh hưởng đến hoạt động của vi điều khiển. Ngoài ra trên đây còn được tích hợp thêm cơ chế cách ly bởi PC817 để tách khối vi điều khiển hẳn ra với khối này.

- 3 là khối cảm biến, khối này dùng các header ra chân để cảm các cảm biến thực tế vào, trong dự án dùng 4 loại cảm biến là cảm biến độ đục, nhiệt độ, pH và cảm biến khoảng cách. Khối này được đặt ở rìa board để đảm bảo dễ dàng kết nối các cảm biến vào. Các đường dây tín hiệu của khối này được đi khác lớp với dây nguồn, nhất là nguồn 5v để giảm nhiễu điện áp gây sai lệch tín hiệu và khối này cũng được đặt xa khối điều khiển cơ cấu chấp hành nhất có thể.
- 4 là header ra chân cho ST-Link để nạp code cho vi điều khiển STM32F103C8T6. Khối này được đặt ở rìa board để dễ dàng kết nối với ST-Link và nạp code.
- 5 là nút reset cứng cho vi điều khiển STM32F103C8T6.
- 6 là khối nguồn cho toàn bộ tầng thiết bị. Khối này dùng USB Type loại 6 chân để cấp nguồn vào tầng thiết bị, với nguồn 3.3V được lấy qua AMS1117-3.3V cấp cho vi điều khiển STM32 và các cảm biến dùng 3.3V, đồng thời nguồn 5V lấy từ chân in của ic nguồn sẽ cấp cho chân 5V trên ESP32 Wrom32 Module (Vì trên đây đã có ic rồi) và cấp cho các cảm biến, relay dùng nguồn 5V.

Qua quá trình thi công và kiểm thử, hệ thống phần cứng đã được triển khai đầy đủ, đảm bảo khả năng thu thập dữ liệu, xử lý tín hiệu và điều khiển thiết bị một cách chính xác và ổn định. Các khối chức năng như vi điều khiển STM32, module ESP32, hệ thống relay, cảm biến và nguồn cấp đều hoạt động đúng theo thiết kế.

## 4.2 KẾT QUẢ PHẦN MỀM

### 4.2.1 Kết quả phần mềm tầng ứng dụng

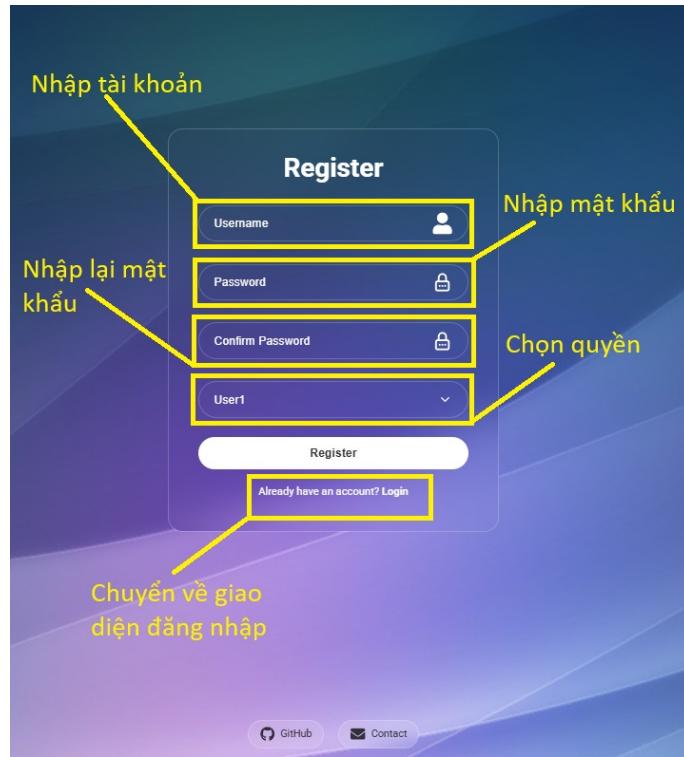
#### 4.2.1.1 Kết quả thiết kế giao diện đăng nhập và đăng ký

Các hình dưới đây mô tả chi tiết về giao diện đăng nhập:



Hình 4.3: Giao diện đăng nhập

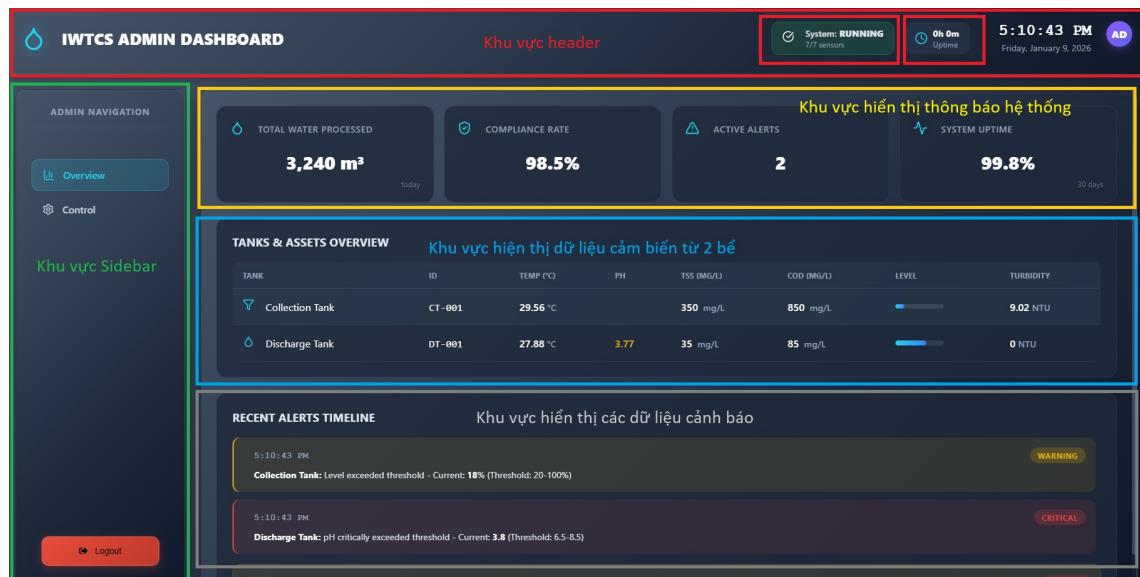
Tại giao diện đăng nhập, nếu người dùng đã có tài khoản thì nhập thông tin vào và tiếp tục login vào để thực hiện điều khiển và giám sát hệ thống



Hình 4.4: Giao diện đăng ký

Tại giao diện đăng ký, nơi đây dành cho người dùng chưa có tài khoản vào để tạo tài khoản, có ba quyền để chọn là Admin, User1 và User2.

#### 4.2.1.2 Kết quả thiết kế giao diện quyền Admin

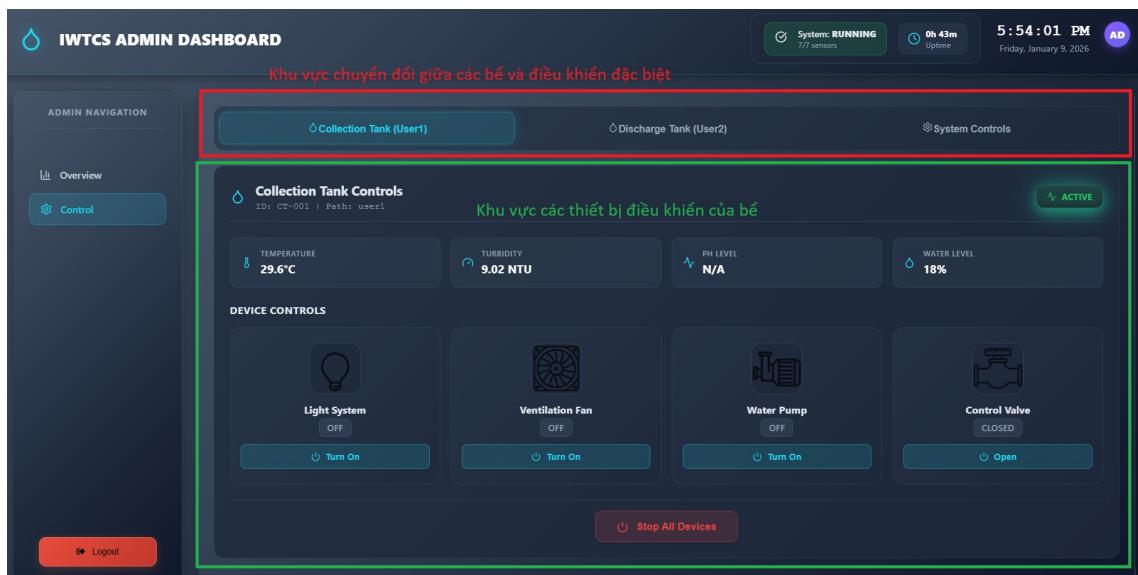


Hình 4.5: Giao diện giám sát với quyền Admin

Giao diện giám sát với quyền Admin này được chia thành các khu vực cụ thể:

- Khu vực header: sẽ hiển thị các thành phần gồm tên hệ thống, tổng các cảm biến trong hệ thống, thời gian hệ thống đã hoạt động.
- Khu vực Sidebar: sẽ là nơi để chuyển đổi qua lại giữa giao diện giám sát và giao diện điều khiển.
- Khu vực hiển thị thông báo hệ thống: trong khu vực này chỉ có cảnh báo là thực (active alerts) còn lại các con số kia là giả lập.
- Khu vực hiển thị dữ liệu cảm biến từ hai bể: sẽ hiển thị tổng thể các dữ liệu cảm biến thu thập được từ hai bể cho người quản trị quan sát và đánh giá.
- Khu vực hiển thị các cảnh báo: khi dữ liệu vượt ngưỡng cài đặt thì các cảnh báo sẽ xuất hiện ở đây tùy theo mức độ: màu vàng là cảnh báo nhẹ, màu đỏ là cảnh báo mạnh.

Tiếp theo là giao diện điều khiển:



Hình 4.6: Giao diện điều khiển với quyền Admin

Tại giao diện điều khiển, người quản trị có thể chuyển đổi qua lại giữa các bể để thực hiện điều khiển các thiết bị trong hệ thống và đồng thời cũng có thể chuyển

đến System Controls để thực hiện các thao tác điều khiển đặc biệt như tắt toàn bộ thiết bị khi khẩn cấp, bật chế độ auto cho các bể (chế độ auto ở đây được thiết lập đơn giản là nếu nước dưới 30% thì bể ngưng bơm, tránh làm cạn bể và nếu nước trên 70% thì bể bắt đầu bơm nước qua bể khác trong quy trình).

#### 4.2.1.3 Kết quả thiết kế giao diện quyền User1



Hình 4.7: Giao diện giám sát và điều khiển quyền User1

Giao diện bao gồm các thành phần sau:

- Khu vực sidebar: tại đây sẽ hiển thị username của người dùng, quyền đăng nhập, hai nút chuyển đổi qua lại giữa điều khiển và giám sát hoặc xem các biểu đồ dữ liệu theo thời gian thực.
- Khu vực hiển thị các thông báo: hiển thị ngày giờ, nhiệt độ,... tại khu vực (Cái này từ API thời tiết cung cấp) và khung hiển thị các dữ liệu đã được gửi đi để truy vết truy cập hệ thống.
- Khu vực điều khiển: giúp người dùng với quyền User1 sẽ điều khiển các thiết bị tại bể thu gom.
- Khu vực điều khiển đặc biệt: cung cấp hai chế độ là chế độ auto và tắt tất cả các thiết bị khi xảy ra sự cố nghiêm trọng.

- Khu vực hiển thị dữ liệu cảm biến: hiển thị các thông số cảm biến thu thập được tại bể thu gom.

Sau khi chuyển qua Chart từ sidebar ta được giao diện như sau:



Hình 4.8: Giao diện hiển thị các biểu đồ dữ liệu quyền User1

#### 4.2.1.4 Kết quả thiết kế giao diện quyền User2

Do giao diện của User2 có cấu trúc và thành phần tương tự như User1, nên trong phần này chỉ tập trung trình bày các kết quả giao diện đã được thiết kế thay vì mô tả chi tiết. Cụ thể các kết quả đạt được trình bày dưới đây:



Hình 4.9: Giao diện giám sát và điều khiển quyền User2



Hình 4.10: Giao diện hiển thị các biểu đồ dữ liệu quyền User2

#### 4.2.2 Kết quả phần mềm tầng gateway

Các bước cấu hình ban đầu: trước tiên là chạy file kernel module nf\_kernel.ko và kiểm tra xem kernel đã được load vào hay chưa bằng lệnh: sudo insmod nf\_kernel.ko sau đó dùng lệnh lsmod để kiểm tra.

```
pi@raspberrypi: ~/MyUser/kernel
File Edit Tabs Help
pi@raspberrypi:~/MyUser/kernel $ ls
Makefile      Module.symvers  nf_kernel.ko  nf_kernel.mod.c  nf_kernel.o
modules.order  nf_kernel.c   nf_kernel.mod  nf_kernel.mod.o
pi@raspberrypi:~/MyUser/kernel $ sudo insmod nf_kernel.ko
pi@raspberrypi:~/MyUser/kernel $ ls | grep nf_kernel.ko
nf_kernel.ko
pi@raspberrypi:~/MyUser/kernel $ lsmod
Module           Size  Used by
nf_kernel        12288  0
snd_seq_dummy    12288  0
snd_hrtimer      12288  1
snd_seq          86016   7 snd_seq_dummy
snd_seq_device   16384   1 snd_seq
uhid             16384   1
rfcomm            57344   6
cmac              12288   3
algif_hash       12288   1
aes_arm64        12288   3
aes_generic      32768   1 aes_arm64
algif_skcipher   12288   1
af_alg            28672   6 algif_hash,algif_skcipher
bnep              24576   2
binfmt_misc      16384   1
brcmfmac_wcc     12288   0
```

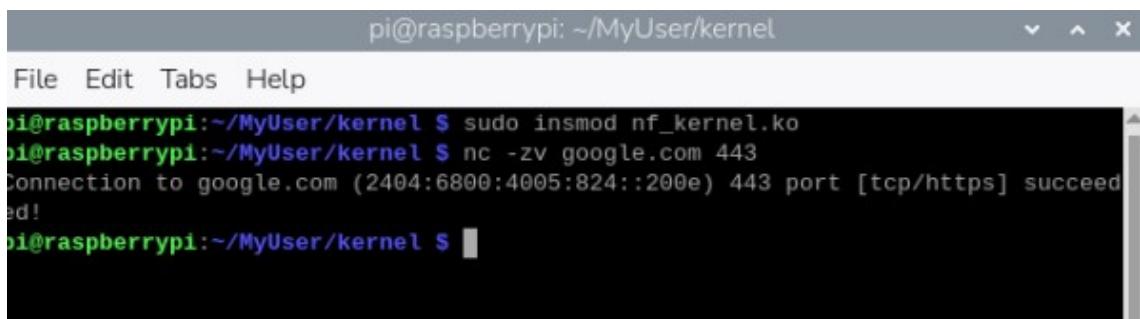
Hình 4.11: Load kernel và kiểm tra ban đầu

Tiếp theo, hệ thống tiến hành chạy các script để xử lý luồng dữ liệu đầu vào,

bao gồm hai file get-info.py và parse\_new\_ver.py. Hai file này được đóng gói trong một bash shell script có tên run\_process.sh. Để thực thi, trước tiên cần phân quyền bằng lệnh chmod, sau đó chạy script bằng cú pháp ./run\_process.sh. Script này đảm nhiệm việc lấy thông tin từ Firestore khi có người dùng mới đăng ký, lưu dữ liệu vào SQLite để phục vụ quá trình xác thực, đồng thời tách gói tin và chuyển về cho ESP32 qua giao thức MQTT nếu người dùng có quyền hợp lệ và kernel module cho phép.

Sau khi chạy xong run\_process.sh, hệ thống sẽ tiến hành thử nghiệm kết quả xử lý tại tầng gateway. Bước đầu tiên là kiểm tra hiệu quả của kernel module ở đầu ra, cụ thể là thử truy cập ra bên ngoài với các port đã bị cấm để đánh giá khả năng kiểm soát truy cập. Với kernel module này truy cập đầu ra sẽ không cấm port 443 (HTTPS) ở đầu ra, nên chỗ này ta thử nc -zv google.com 443 khi đó kết quả sẽ thấy connected ngay lập tức.

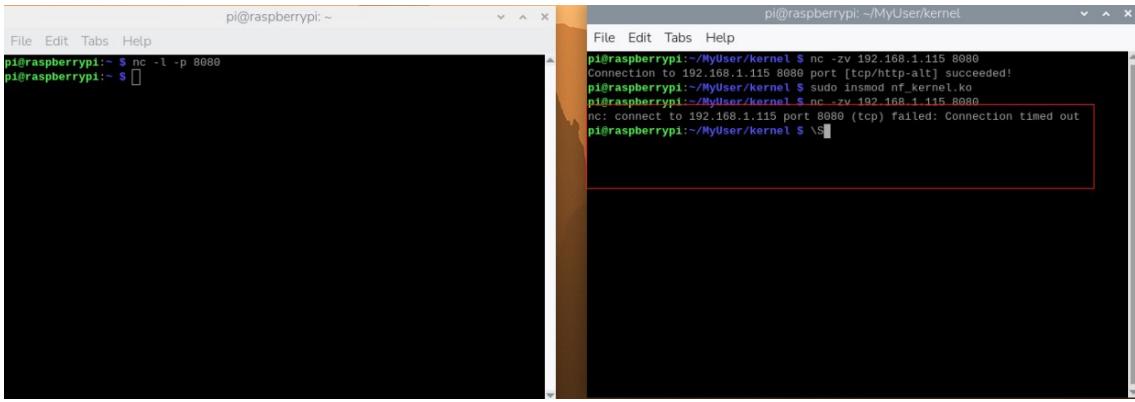
Dưới đây là các bước thử nghiệm chi tiết và kết quả:



```
pi@raspberrypi:~/MyUser/kernel
File Edit Tabs Help
pi@raspberrypi:~/MyUser/kernel $ sudo insmod nf_kernel.ko
pi@raspberrypi:~/MyUser/kernel $ nc -zv google.com 443
Connection to google.com (2404:6800:4005:824::200e) 443 port [tcp/https] succeeded!
pi@raspberrypi:~/MyUser/kernel $
```

Hình 4.12: Kiểm thử đầu ra kernel module

Ngược lại ở đầu vào để kiểm thử, ta sẽ dùng hai terminal để kiểm tra thử mở kết nối port 8080. Khi này, nếu chưa có kernel module được load thì sẽ thành công ngay lập tức, ngược lại, nếu như đã load kernel module rồi thì sẽ bị chờ, timeout hoặc refused, cụ thể kết quả sẽ được trình bày bên dưới:



Hình 4.13: Kiểm thử vào kernel module

Test luồng điều khiển trên gateway: ở đây sẽ thực hiện các bước để test luồng điều khiển được xử lý như thế nào trên gateway khi có lệnh điều khiển từ người dùng với 2 trường hợp: hợp lệ và không hợp lệ. Trước tiên dùng tài khoản có quyền User1 là thai\_u1 để kiểm tra bật thử một thiết bị trên giao diện và xem kết quả:

```
Collection: terminal_commands
Processing command: db/thai_u1/user1/bulb1/on
User: thai_u1
Type: input
Parsed:
  - Username: thai_u1
  - Role: user1
  - Device: bulb1
  - Action: on
User 'thai_u1' has permission 'user1'
✓ User 'thai_u1' được xác thực với quyền 'user1'
Sent to MQTT: bulb1/on
SUCCESS: Command processed and sent to ESP32
Message published to MQTT (mid: 17)
```

Hình 4.14: Kiểm thử luồng điều khiển

Khi này do tài khoản đã được đăng ký nên người dùng này có được quyền điều khiển và gói tin sẽ được đưa vào tầng thiết bị xử lý, sau đó relay được bật.

```
Collection: terminal_commands
Processing command: db/guest/user/valve1/on
User: guest
Type: input
Parsed:
  - Username: guest
  - Role: user
  - Device: valve1
  - Action: on
User 'guest' didn't exist SQLite
DENIED: User 'guest' không tồn tại trong database
```

Hình 4.15: Kiểm thử luồng điều khiển tiếp theo

Khi một tài khoản chưa được đăng ký thì sẽ không được quyền gửi dữ liệu qua MQTT, khi đó gói tin điều khiển sẽ bị hủy ngay lập tức và không có thay đổi trạng thái trên phần cứng.

Về luồng dữ liệu gửi đi chính là dữ liệu cảm biến, ta thực hiện chạy file firebase\_rt.py và nhận được kết quả:

```
Connected to MQTT broker
Subscribed to: iot/sensor1
Subscribed to: iot/sensor2

Received from iot/sensor1 → user1:
Data: {
temp": 26,
water": 18,
tds": 0

user1/temperature/history/1767998247/temp: 26
user1/temperature/current: 26
user1/water_level/history/1767998248/level: 18
user1/water_level/current: 18
user1/turbidity/history/1767998248/turbidity: 0
user1/turbidity/current: 0
user1 data logged successfully
```

Hình 4.16: Kiểm thử luồng dữ liệu gửi đi

```
Received from iot/sensor2 → user2:
Data: {
temp": 26.06,
water": 63,
tds": 0,
ph": 5.54

user2/temperature/history/1767998250/temp: 26.06
user2/temperature/current: 26.06
user2/water_level/history/1767998251/level: 63
user2/water_level/current: 63
user2/turbidity/history/1767998251/turbidity: 0
user2/turbidity/current: 0
user2/ph/history/1767998251/ph: 5.54
user2/ph/current: 5.54
user2 data logged successfully
```

Hình 4.17: Kiểm thử luồng dữ liệu gửi đi tiếp theo

Khi này các thông tin dữ liệu cảm biến đã được gửi đi và cập nhật vào Firebase Realtime Database để thực hiện công việc tiếp theo là trực quan hóa dữ liệu.

### 4.2.3 Kết quả phần mềm tầng thiết bị

Trong quá trình triển khai, phần mềm tầng thiết bị đóng vai trò quan trọng trong việc thu thập dữ liệu từ các cảm biến, xử lý tín hiệu ban đầu và truyền thông tin đến tầng gateway. Các chương trình được xây dựng trên vi điều khiển STM32 và module ESP32 nhằm đảm bảo khả năng đọc dữ liệu chính xác, phản hồi nhanh và giao tiếp ổn định với các thành phần khác trong hệ thống. Phần này sẽ trình bày kết quả thực nghiệm của phần mềm tầng thiết bị, bao gồm quá trình khởi tạo, thu thập dữ liệu cảm biến, xử lý tín hiệu và truyền thông qua giao thức UART/MQTT, qua đó đánh giá mức độ đáp ứng của hệ thống so với yêu cầu thiết kế.

```
06:05:59.950 -> WiFi OK
06:05:59.950 -> MQTT connecting... OK
06:06:00.110 -> STM32 UART1 -> SENSOR1:TEMP:25.94;WATER:0;TDS:0.00;PH:0.00
06:06:00.146 -> MQTT publish -> SENSOR1 (water: 0cm → 100%): {"temp":25.94,"water":100,"tds":0}
06:06:00.793 -> STM32 UART1 -> SENSOR2:TEMP:25.88;WATER:0;TDS:0.00;PH:5.29
06:06:00.793 -> MQTT publish -> SENSOR2 (water: 0cm → 100%): {"temp":25.88,"water":100,"tds":0,"ph":5.29}
06:06:03.392 -> STM32 UART1 -> SENSOR1:TEMP:25.94;WATER:11;TDS:0.00;PH:0.00
06:06:03.392 -> MQTT publish -> SENSOR1 (water: 11cm → 9%): {"temp":25.94,"water":9,"tds":0}
06:06:04.956 -> STM32 UART1 -> SENSOR2:TEMP:25.88;WATER:5;TDS:0.00;PH:5.32
06:06:04.956 -> MQTT publish -> SENSOR2 (water: 5cm → 63%): {"temp":25.88,"water":63,"tds":0,"ph":5.32}
```

Hình 4.18: Kết quả phần mềm trên tầng thiết bị

Ở tầng thiết bị, bước khởi tạo kết nối bao gồm kết nối Wifi và MQTT, MQTT nếu không tìm thấy broker thì nó hoàn toàn không gửi thông tin dữ liệu đi để đảm bảo an toàn dữ liệu.

Tiếp theo là khi nhận được dữ liệu từ tầng gateway:

```
06:22:05.769 -> MQTT publish -> SENSOR1 (water: 11cm → 9%): {"temp":25.88,"water":9,"tds":0}
06:22:06.064 -> TX -> STM32 (UART2): pump1/on
06:22:07.593 -> TX -> STM32 (UART2): pump1/off
06:22:08.700 -> STM32 UART1 -> SENSOR2:TEMP:25.75;WATER:5;TDS:0.00;PH:5.75
06:22:08.732 -> MQTT publish -> SENSOR2 (water: 5cm → 63%): {"temp":25.75,"water":63,"tds":0,"ph":5.75}
06:22:09.025 -> TX -> STM32 (UART2): pump1/on
06:22:09.902 -> STM32 UART1 -> SENSOR1:TEMP:25.88;WATER:11;TDS:0.00;PH:0.00
06:22:09.902 -> MQTT publish -> SENSOR1 (water: 11cm → 9%): {"temp":25.88,"water":9,"tds":0}
06:22:12.858 -> STM32 UART1 -> SENSOR2:TEMP:25.75;WATER:5;TDS:0.00;PH:5.75
06:22:12.858 -> MQTT publish -> SENSOR2 (water: 5cm → 63%): {"temp":25.75,"water":63,"tds":0,"ph":5.75}
06:22:14.064 -> STM32 UART1 -> SENSOR1:TEMP:25.88;WATER:11;TDS:0.00;PH:0.00
06:22:14.064 -> MQTT publish -> SENSOR1 (water: 11cm → 9%): {"temp":25.88,"water":9,"tds":0}
06:22:17.023 -> STM32 UART1 -> SENSOR2:TEMP:25.75;WATER:5;TDS:0.00;PH:5.76
```

Hình 4.19: Kết quả phần mềm trên tầng thiết bị tiếp theo

Khi một gói tin hợp lệ được gateway duyệt thì ESP32 trên tầng thiết bị sẽ nhận và chuyển liền cho STM32 qua UART2.

Sau quá trình triển khai, mô hình IoT bảo mật đa tầng đã hoàn thiện đầy đủ cả phần cứng và phần mềm với khả năng giám sát thời gian thực, điều khiển từ xa và phân quyền truy cập rõ ràng. Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, dữ liệu cảm biến được thu thập và hiển thị liên tục, độ trễ thấp và các lệnh điều khiển được thực thi chính xác trong phạm vi phân quyền của người dùng. Cơ chế bảo mật đa tầng, bao gồm xác thực người dùng, phân quyền tại tầng ứng dụng và kiểm soát lệnh điều khiển tại tầng kernel thông qua Netfilter và SQLite, đã phát huy hiệu quả trong việc hạn chế truy cập trái phép và ngăn chặn thao tác sai quy trình.

Xét trên các yêu cầu kỹ thuật đề ra, hệ thống đáp ứng tốt các tiêu chí về giám sát thông số môi trường, điều khiển thiết bị chấp hành, phân chia vai trò người dùng và đảm bảo an toàn vận hành trong môi trường có kết nối mạng. Nhìn chung, mô hình đạt tính ổn định, khả năng mở rộng cao và có tiềm năng ứng dụng trong đào tạo, nghiên cứu và định hướng triển khai trong hệ thống xử lý nước thải thực tế.

# **CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỀN**

## **5.1 KẾT LUẬN**

Đề tài đã nghiên cứu, thiết kế và triển khai thành công mô hình IoT bảo mật đa tầng ứng dụng trong quy trình xử lý nước thải công nghiệp. Mô hình được xây dựng theo kiến trúc phân tầng gồm tầng thiết bị, tầng gateway và tầng ứng dụng, trong đó yếu tố bảo mật và phân quyền được tích hợp xuyên suốt hệ thống. Các kết quả thực nghiệm cho thấy cơ chế bảo mật đa tầng không chỉ ngăn chặn truy cập trái phép từ phía người dùng mà còn hạn chế nguy cơ tấn công trực tiếp vào hệ thống điều khiển nhờ lớp lọc tại kernel-space trên gateway.

Hệ thống đáp ứng tốt yêu cầu về giám sát thời gian thực, kiểm soát chấp hành và phân quyền theo vai trò người dùng (Admin – User1 – User2). Đồng thời, việc kết hợp FreeRTOS trên tầng thiết bị, MQTT trên tầng truyền thông và Firebase trên tầng ứng dụng đã cho phép hệ thống duy trì khả năng vận hành ổn định, dữ liệu đồng bộ và giao diện trực quan, hỗ trợ giám sát – điều khiển linh hoạt.

## **5.2 ĐÁNH GIÁ CHUNG**

Về mặt kỹ thuật, đề tài đạt được các mục tiêu chính đã đề ra, bao gồm:

- Kiến trúc phần cứng và phần mềm rõ ràng.
- Cơ chế bảo mật đa tầng hoạt động hiệu quả, kết hợp xác thực, phân quyền và lọc lệnh điều khiển.
- Dữ liệu cảm biến ổn định, độ trễ thấp, phù hợp yêu cầu giám sát môi trường.
- Giao diện người dùng trực quan, phân tách quyền hợp lý và đảm bảo an toàn vận hành.

Mặc dù mô hình mới dừng lại ở quy mô phòng thí nghiệm, kết quả triển khai

cho thấy tính khả thi cao nếu mở rộng sang hệ thống xử lý nước thải công nghiệp thực tế.

### 5.3 HẠN CHẾ

Một số hạn chế của đề tài bao gồm:

- Chưa tích hợp đầy đủ các thông số môi trường như COD/BOD, TSS hoặc các cảm biến nâng cao.
- Cơ chế giám sát dữ liệu mới dừng ở cấp hiển thị trực quan, chưa khai thác phân tích dữ liệu hoặc dự báo.
- Chưa tích hợp hệ thống ghi log vận hành và cảnh báo sự kiện an toàn ở quy mô công nghiệp.
- Bộ kit mô phỏng chưa phản ánh đầy đủ điều kiện môi trường thực tế như nhiễu, sai số đo hoặc tải hệ thống.

### 5.4 HƯỚNG PHÁT TRIỂN

Trong tương lai, đề tài có thể được mở rộng theo hướng bổ sung thêm các thông số môi trường nâng cao và mở rộng đối tượng cảm biến, đồng thời tích hợp các cơ chế phân tích dữ liệu, cảnh báo và dự đoán dựa trên học máy nhằm hỗ trợ vận hành thông minh hơn. Bên cạnh đó, hệ thống có thể hoàn thiện các chức năng giám sát an toàn, nhật ký vận hành và theo dõi truy cập, đi kèm với việc nâng cấp hạ tầng bảo mật theo chuẩn IIoT và các tiêu chuẩn công nghiệp như ISA/IEC 62443. Ở cấp độ tích hợp, mô hình có thể triển khai trên quy mô thực tế với giao tiếp công nghiệp (ví dụ Modbus), cũng như kết nối dữ liệu với các hệ thống quản lý vận hành như SCADA nhằm phục vụ mục tiêu chuyển đổi số trong công nghiệp.

## **PHỤ LỤC**

Mọi thông tin về mã nguồn chi tiết tại:

<https://github.com/ThaiVM2004/Security-Model-IoT-For-Water-Treatment-System>

Kết quả của sản phẩm được trình bày tại:

<https://www.youtube.com/watch?v=xyskkY6qwX0>

## TÀI LIỆU THAM KHẢO

- [1] T. chí Môi trường, “Thực trạng và đề xuất giải pháp phòng ngừa, ứng phó sự cố chất thải tại một số cơ sở sản xuất có nguy cơ gây ô nhiễm môi trường ở việt nam,” [Online]. Available: <https://tapchimoitruong.vn/chuyen-muc-3/thuc-trang-va-de-xuat-giai-phap-phong-ngua-ung-pho-su-co-chat-thai-tai-mot-so-co-so-san-xuat-co-nguy-co-gay-o-nhiem-moi-truong-o-viet-nam-31370>, 2025, [Accessed: Dec. 27, 2025].
- [2] B. K. học và Công nghệ Việt Nam, “Cẩn trọng khi dùng thiết bị iot,” [Online]. Available: <https://mst.gov.vn/can-trong-khi-dung-thiet-bi-iot-197250105050842715.htm>, 2024, [Accessed: Dec. 27, 2025].
- [3] Y. Zhang, X. Li *et al.*, “Security challenges and solutions in industrial internet of things (iiot),” *Electronics*, vol. 12, no. 12, p. 2590, 2023, [Accessed: Dec. 31, 2025].
- [4] B. Ali, G. Wang *et al.*, “Security and privacy in the industrial internet of things: Current standards and future challenges,” *ResearchGate*, 2020, [Accessed: Dec. 27, 2025].
- [5] R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things,” in *Proceedings of the 2015 International Conference on Computer Communications and Networks (ICCCN)*. ACM, 2015, [Accessed: Dec. 31, 2025].
- [6] T. chí Môi trường, “Chuyển đổi số và thực tiễn ứng dụng trong hoạt động giám sát môi trường,” [Online]. Available: <https://tapchimoitruong.vn/nghien-cuu-23/chuyen-doi-so-va-thuc-tien-ung-dung-trong-hoat-dong-giam-sat-moi-truong-31660>, 2025, [Accessed: Dec. 27, 2025].
- [7] M. T. H. Nhất, “5 quy trình và sơ đồ hệ thống xử lý nước thải thường dùng,” [Online]. Available: <https://moitruonghopnhat.com/5-quy-trinh-va-so-do-he-thong-xu-ly-nuoc-thai-thuong-dung>

so-do-he-thong-xu-ly-nuoc-thai-thuong-dung-2469.html, 2025, [Accessed: Dec. 27, 2025].

- [8] ——, “Quy trình xử lý nước thải công nghiệp,” [Online]. Available: <https://moitruonghopnhat.com/quy-trinh-xu-ly-nuoc-thai-cong-nghiep-313.html>, 2025, [Accessed: Jan. 1, 2026].
- [9] R. P. Foundation, “Introduction to the raspberry pi operating system,” [Online]. Available: <https://www.raspberrypi.com/documentation/computers/os.html#introduction>, 2025, [Accessed: Jan. 1, 2026].
- [10] FreeRTOS, “What is freertos?” [Online]. Available: <https://www.freertos.org/Why-FreeRTOS/What-is-FreeRTOS>, 2025, [Accessed: Jan. 1, 2026].
- [11] A. W. Services, “Announcing freertos kernel v10,” [Online]. Available: <https://aws.amazon.com/blogsopensource/announcing-freertos-kernel-v10/>, 2017, [Accessed: Jan. 1, 2026].
- [12] L. K. Labs, “Introduction to the linux kernel,” [Online]. Available: <https://linux-kernel-labs.github.io/refs/heads/master/lectures/intro.html>, 2025, [Accessed: Jan. 1, 2026].
- [13] Oracle, “Introduction to netfilter,” [Online]. Available: <https://blogs.oracle.com/linux/introduction-to-netfilter>, 2025, [Accessed: Jan. 1, 2026].
- [14] GeeksforGeeks, “Firebase introduction,” [Online]. Available: <https://www.geeksforgeeks.org/firebase/firebase-introduction/>, 2025, [Accessed: Jan. 1, 2026].
- [15] G. Firebase, “Firebase realtime database documentation,” [Online]. Available: <https://firebase.google.com/docs/database>, 2025, [Accessed: Jan. 1, 2026].
- [16] ——, “Firebase authentication documentation,” [Online]. Available: <https://firebase.google.com/docs/auth>, 2025, [Accessed: Jan. 1, 2026].

- [17] ——, “Cloud firestore documentation,” [Online]. Available: <https://firebase.google.com/docs/firestore>, 2025, [Accessed: Jan. 1, 2026].
- [18] S. Tutorial, “Sqlite tutorial,” [Online]. Available: <https://www.sqlitetutorial.net/>, 2025, [Accessed: Jan. 1, 2026].
- [19] HiveMQ, “Mqtt essentials part 1: Introducing mqtt,” [Online]. Available: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>, 2025, [Accessed: Jan. 1, 2026].
- [20] ——, “How to get started with mqtt,” [Online]. Available: <https://www.hivemq.com/blog/how-to-get-started-with-mqtt/>, 2025, [Accessed: Jan. 1, 2026].
- [21] E. Team, “Mosquitto mqtt broker: Pros/cons, tutorial, and modern alternatives,” [Online]. Available: <https://www.emqx.com/en/blog/mosquitto-mqtt-broker-pros-cons-tutorial-and-modern-alternatives>, 2023, [Accessed: Jan. 1, 2026].
- [22] A. Devices, “Uart: A hardware communication protocol,” [Online]. Available: <https://www.analog.com/en/resources/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>, 2025, [Accessed: Jan. 1, 2026].
- [23] Arm, “Introduction to raspberry pi - embedded and microcontrollers learning path,” [Online]. Available: <https://learn.arm.com/learning-paths/embedded-and-microcontrollers/rpi/intro/>, 2025, [Accessed: Jan. 1, 2026].
- [24] E. Systems, “Esp32-wroom-32 datasheet, version 2.9,” [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf), 2019, [Accessed: Jan. 1, 2026].
- [25] R. N. Tutorials, “Getting started with esp32,” [Online]. Available: <https://randomnerdtutorials.com/getting-started-with-esp32/>, 2025, [Accessed: Jan. 1, 2026].

- [26] STMicroelectronics, “Stm32f103xx performance line datasheet,” [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f103c6.pdf>, 2007, rev. 2, Preliminary Data, [Accessed: Jan. 1, 2026].
- [27] D. S. . M. Integrated, “Ds18b20 programmable resolution 1-wire digital thermometer,” [Online]. Available: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>, 2000, preliminary Datasheet, [Accessed: Jan. 1, 2026].
- [28] DFRobot, “Ph meter sku sen0161,” [Online]. Available: [https://wiki.dfrobot.com/ph\\_meter\\_sku\\_\\_sen0161\\_](https://wiki.dfrobot.com/ph_meter_sku__sen0161_), 2025, [Accessed: Jan. 1, 2026].
- [29] ——, “Turbidity sensor sku sen0189,” [Online]. Available: [https://wiki.dfrobot.com/turbidity\\_sensor\\_sku\\_\\_sen0189](https://wiki.dfrobot.com/turbidity_sensor_sku__sen0189), 2025, [Accessed: Jan. 2, 2026].
- [30] S. Electronics, “Hc-sr04 ultrasonic sensor datasheet,” [Online]. Available: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>, 2010, [Accessed: Jan. 2, 2026].